

C O N T E N T S

NR No.	Title	No. of Pages.
<hr/>		
54	BVM Formalism for QL-1 in First Order Predicate Logic.	113
55	Boolean Vector-Matrix Formulation (BVMF) of Logic. Lecture 1 and 2 Series 2.	80
56	Boolean Vector-Matrix Formulation (BVMF) of Logic . Lecture 3 and 4, Series 2.	33
57	Generalization via the theory of relations in BVMF of the clausal form of Logic and its applications.	47
58	Symmetry properties of Logical Functions in Propositional Calculus.	37
59	Tensor Algebra in BVMF.	13

BVM Formalism for QL-1 in First Order Predicate Logic

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

vi. Formalism for QL-1 in First Order Predicate Logic

G.N. Ramachandran

IRSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

CONTENTS

	Page No
1. Introduction — QL-1A matrix connectives	1
2. EVM formulation of QL-1 connectives AND, OR, EQU represented by the symbols \underline{A} , \underline{O} , \underline{E} :	5
(a) Definition of the matrix connectives $\underline{Z}(k, \ell)$	5
(b) Outline of the theory of QL-1A matrix connectives	7
(c) Implementation of QL-1A unary and binary relations	16
(i) Binary relation in QL-1A	16
(ii) Unary relation in QL-1A	17
3. Implementation of the QL-1 binary relation	19
4. Implementation of the QL-1 unary relation	23
(a) Formulae for the standard quantifiers $\forall, \exists, \wedge, \Phi$ (IS-1 formulation <u>via</u> lattice algebra)	23
(b) Formulation for standard quantifier states using matrix algebra and QL-1 connectives \underline{Z}	25
(c) QL-1 binary reverse relation leading to unary relation	31
5. QBA implementation of QL-1 unary relations	32
(a) Statement of the problem	32
(b) QL-1A binary reverse relation for the generators of QBA	34
(c) QL-1 unary relation for all eight states of QBA	41
(i) Logical checking of Table 6(a) for \underline{A}	42
(ii) Comments on Tables 6(b,c,d) for \underline{O} , \underline{I} and \underline{E}	48
(iii) Comments on the right hand half of Tables 6(a-d) and occurrence of two inconsistencies in each	49
(d) Implementation of QL-1 unary relations for $\underline{Z}(k, \ell)$	52
(i) Consistency check of the input against the relation and its consequent modification	53

CONTENTS

Page No

6. Multiply quantified relations in QL-1	54
(a) Unary relation	55
(b) Binary forward relation	57
(c) Interconversion of QL-1 and QL-2 forms of relations in predicate logic	59
(i) Case with one term without variable	59
(ii) Case with both terms having two different variables.	68
(iii) Interconversion of QL-1 and QL-2 forms of statements in predicate logic	69
(d) Interconversion of QL-1 and QL-2 forms of relations in predicate logic (revised version of Section c)	72
(i) Generalization of the standard relations of this type employed for the prenex normal form	72
(ii) Table of equivalent QL-1 and QL-2 forms of (78a,b)	75
(iii) Proof of the QL-1 — QL-2 transformation equations for the standard quantifier states	
(iv) A new inconsistency theorem for the standard formulation of first order predicate logic	94
(v) Re-examination of QL-1,2 transformation equations for connective E	98
(e) QL-1 — QL-2 interconversion of statements in general	104
(i) Implementation of the transformation	106
(ii) Implementation of the transformation from the QL-1 to the QL-2 form	107
(iii) Examples of interconversion formulae	109

BVM Formalism for QL-1 in First Order Predicate Logic

1. Introduction — QL-1A matrix connectives

While writing Section 7 on the QL-1 formalism for MR-53, it was noticed that what had been worked out earlier in IS-1 and ALOG-48 regarding this were not quite satisfactory and many aspects of the theory for QL-1 had to be worked out and a fuller theory developed. Therefore, the whole subject of QL-1 is re-examined here. Starting from an elementary relation in one variable of the type in Eq.(1) below, the theory is extended to compound sentences quantified by $(q_z x)$ of the type in Eq.(2) and also to elementary statements having multiple quantifiers. The notation for quantifier states will be borrowed from MR-53 as also the theory of the BA-3 algebra of QBA.

The format for an elementary statement in QL-1, written in standard notation, is :

$$(q_z x)(\underline{a}x \underline{Z}(k, \ell) \underline{b}x), (q_z x) = q(i), i = 1 \text{ to } 8, k, \ell = 1, 2 \quad (1a)$$

$$\text{e.g. } (\forall x)(\underline{a}x \implies \underline{b}x) \equiv q(\underline{1}Zx) (\underline{a}x \underline{I}(1,1) \underline{b}x) \quad (1b)$$

In BVMF notation, for QL-1, this takes the form of Eq.(2):

$$\underline{a}x \underline{Z} \underline{b}x, \text{ where } \underline{a}, \underline{b} \text{ are dummy symbols and } \underline{Z} = (\underline{q}_Z, \underline{Z}(k, \ell)) \quad (2, a, b)$$

We shall use the same symbols $\underline{a}x, \underline{b}x$ for quantified terms as in QL-2 (since they have the same significance) and only denote the QL-1 relation by the symbol \underline{Z} as shown in Eq.(2). Clearly, \underline{q}_Z of Eq.(2b) is one of the eight quantifier states, and $\underline{Z}(k, \ell)$ is the SNS matrix connective in the relation $\underline{a}x \underline{Z} \underline{b}x$, which is the predicate in Eq.(1a).

The binary logical QL-1 relation in Eq.(2) can be implemented either in the unary form or binary form as with binary logical relations in SNS and QL-2. Thus, in standard notation, denoting \underline{Z} by $\underline{Z}(k, \ell)$, the implementation of these two forms is indicated, in principle, in Eqs. (3) and (4) below:

Unary relation

$$(\underline{q}_a, x)(\underline{a}'x), (\underline{q}_Z x)(\underline{a}x \underline{Z} \underline{b}x) \mapsto (\underline{q}_b, x)(\underline{b}'x) \quad (3a)$$

and e.g.

$$\underline{a}'x = (\exists x)(T), (\forall x)(\underline{a}x \Rightarrow \underline{b}x) \mapsto \underline{b}'x = (\exists x)(T) \quad (3b)$$

Binary relation

$$(\underline{q}_a, x)(\underline{a}'x), (\underline{q}_b, x)(\underline{b}'x), (\underline{q}_Z x)(\underline{a}x \underline{Z} \underline{b}x = \underline{c}x) \mapsto \underline{t}(\underline{a} \underline{Z} \underline{b} \mid \underline{a}', \underline{b}') \quad (4a)$$

and $(\underline{q}_c, x)(\underline{c}'x)$

$$\text{e.g. } \underline{a}'x = (\forall x)(F), \underline{b}'x = (\exists x)(T); (\forall x)(\underline{a}x \wedge \underline{b}x = \underline{c}x) \mapsto \underline{t}(\underline{Z} \mid \underline{a}', \underline{b}') = D, \underline{c}'x = (\exists x)(F) \quad (4b)$$

Thus, the output is a quantified term for the unary relation (3a) while it can be obtained either as a quantified term $\underline{c}'x$ or as an SNS term for the truth value of the binary relation corresponding to the given inputs $\underline{a}'x$ and $\underline{b}'x$ as in (4a).

In the following Sections 2, 3 and 4, we shall establish formulae that are necessary for implementing unary and binary forms of statements composed of elementary relations in QL-1. This is done by first considering a singly quantified elementary relation in QL-1 for the four standard QL-1 states \forall , \exists , \wedge and \neg . For these states, the lattice number notation given in ALOG-48 and IS-1 is very useful and will be used where necessary, although the notation of QL-2 for quantifiers is the one adopted in the formalism developed in this report. For ready reference, and also for application in the theory of what is termed as QL-1A relations below, we need a proper definition of the eight quantifier states in QBA and their description, via set theory and the truth values of the individual constituent term \underline{ax} . Therefore, we reproduce Table 1 from ALOG-50 (p.4) and it contains all the necessary definitions and descriptions of the eight QBA states $q(1)$ to $q(8)$ as well as their representation in BA-3 algebra.

Table 1: Properties of the quantifier Boolean algebra with
SNS truth values

Sl. No.	Quantifier state \underline{q}			Truth value of $\bigoplus_x \underline{ax}^*$		
	Name	Symbol	Set-theoretical description	Description in BA-2	Symbol in BA-3 †	3-vector
q(1)	For all	\forall	All $n \underline{ax}$ are T	Always T	TT (Gen)	(1 0 0)
q(3)	For some	\exists	1 to $n-1 \underline{ax}$ are T, 1 to $n-1 \underline{ax}$ are F, and the rest are D.	D, but not T and not F	SS (Gen)	(0 1 0)
q(5)	For none	\emptyset	All $n \underline{ax}$ are F	Always F	FF (Gen)	(0 0 1)
q(6)	There exists	\exists	1 to $n \underline{ax}$ are T and the rest are D or F.	D or T, but not F	$ET = SS \oplus TT$	(1 1 0)
q(4)	All or none	\ominus	All $n \underline{ax}$ are T or all $n \underline{ax}$ are F.	T or F, but not D	$TF = TT \oplus FF$	(1 0 1)
q(2)	Not for all	\wedge	1 to $n \underline{ax}$ are F, and the rest are D or T.	D or F, but not T	$EF = SS \oplus FF$	(0 1 1)
q(7)	Indefinite	\triangle	1 to $n \underline{ax}$ may be T, F, or D.	D, may also be T or F	$DD = TT \oplus SS \oplus FF$	(1 1 1)
q(8)	Impossible	ϕ	1 to $n \underline{ax}$ are X, others may be T, F or D.	Not D, not T and not F	$XX = TT \otimes SS \otimes FF$	(0 0 0)

* $\bigoplus_x \underline{ax}$ is the logical representation of the set-theoretical description. Although its description is in BA-2, its range is not a truth value of BA-2 and requires BA-3 algebra.

† Gen = generator.

It is found that the implementation of Eq.(4a), in general, requires a new way of applying the SNS connectives $\underline{\underline{A}}$, $\underline{\underline{O}}$, $\underline{\underline{E}}$ between two quantified terms (both in x) of the form (5a,b,c):

$$\underline{\underline{a}}x \underline{\underline{A}} \underline{\underline{b}}x = \underline{\underline{c}}x ; \quad \underline{\underline{a}}x \underline{\underline{O}} \underline{\underline{b}}x = \underline{\underline{c}}x ; \quad \underline{\underline{a}}x \underline{\underline{E}} \underline{\underline{b}}x = \underline{\underline{c}}x \quad (5a,b,c)$$

Unlike the SNS binary relation $\underline{\underline{a}} \underline{\underline{Z}}(k, \ell) \underline{\underline{b}} = \underline{\underline{c}}$, and the QL-2 relation $\underline{\underline{a}} \underline{\underline{Z}}(k, \ell) \underline{\underline{b}} = \underline{\underline{c}}$, for both of which the output is an SNS vector $\underline{\underline{c}}$, in the case of Eqs (5a,b,c), the output is also a quantified term in QL-1, namely $\underline{\underline{c}}x$. The theory of the connectives in Eqs. (5a,b,c) is given in the next section 2, after which the implementation of QL-1 binary and unary relations are discussed in Sections 3 and 4.

As an example, the application of the QL-1 connective $\underline{\underline{A}}$ may be given as follows:

$$(\forall x)(\underline{\underline{a}}x) \underline{\underline{A}} (\exists x)(\underline{\underline{b}}x) = (\exists x)(\underline{\underline{a}}x \underline{\underline{A}} \underline{\underline{b}}x) = (\exists x)(\underline{\underline{c}}x) \quad (6a)$$

which has clearly the form

$$\underline{\underline{a}}x \underline{\underline{A}} \underline{\underline{b}}x = \underline{\underline{c}}x \quad (6b)$$

The type of relations appearing in Eqs. (5a,b,c) will be denoted as QL-1A type of relation, and its properties with

regard to the nature of the output have been mentioned above. In fact, just as SNS and QL-2 logical relations which also have a unary form of implementation, we can also think of the unary form for the three connectives contained in Eqs. (5a,b,c) which are as follows:

$$\underline{ax} \underline{Z} = \underline{bx} \quad , \quad \underline{Z} = \underline{A}, \underline{O}, \underline{E} \quad (7)$$

Both the unary and binary forms of implementation will become clear from the formal definition of the QL-1 relations which is given in the next Section 2(a).

2.- BVM formulation of QL-1 connectives AND, OR, EQU, represented by the symbols \underline{A} , \underline{O} , \underline{E} .

(a) Definition of the matrix connectives $\underline{Z}(k, \ell)$.

We suppose that the set $\{x\}$ has values from 1 to n, and that $\{\underline{ax}\}$ and $\{\underline{bx}\}$ are representable by a set of N SNS truth values as in (8a,b):

$$\underline{ax} \equiv (\underline{a1}, \underline{a2}, \dots, \underline{an}) ; \quad \underline{bx} = (\underline{b1}, \underline{b2}, \dots, \underline{bn}) \quad (8a,b)$$

The (2,n)-vectors composed of \underline{ax} and \underline{bx} can be denoted by the quantified terms \underline{ax} and \underline{bx} respectively, these symbols representing their QBA states (as described in MR-53).

However, for the QL-1A formalism, it is not sufficient to know the QBA state as such, but it is also necessary to have the full details about the $(2, n)$ -vectors denoting the two quantified terms \underline{a}_x and \underline{b}_x (as given in column 4 of Table 1). In terms of these, we define the connectives \underline{A} , \underline{Q} and \underline{E} of QL-1A as in (9a,b,c), in terms of the corresponding SNS connectives $\underline{\underline{A}}$, $\underline{\underline{Q}}$, $\underline{\underline{E}}$ respectively.

$$(\underline{a}_x \underline{A} \underline{b}_x = \underline{c}_x) \equiv (\underline{a}_1 \underline{\underline{A}} \underline{b}_1 = \underline{\underline{c}}_1, \dots, \underline{a}_n \underline{\underline{A}} \underline{b}_n = \underline{\underline{c}}_n) \quad (9a)$$

$$(\underline{a}_x \underline{Q} \underline{b}_x = \underline{c}_x) \equiv (\underline{a}_1 \underline{\underline{Q}} \underline{b}_1 = \underline{\underline{c}}_1, \dots, \underline{a}_n \underline{\underline{Q}} \underline{b}_n = \underline{\underline{c}}_n) \quad (9b)$$

$$(\underline{a}_x \underline{E} \underline{b}_x = \underline{c}_x) \equiv (\underline{a}_1 \underline{\underline{E}} \underline{b}_1 = \underline{\underline{c}}_1, \dots, \underline{a}_n \underline{\underline{E}} \underline{b}_n = \underline{\underline{c}}_n) \quad (9c)$$

It is to be noted that \underline{A} , \underline{Q} , \underline{E} correspond to $\underline{\underline{A}}(1, 1)$, $\underline{\underline{Q}}(1, 1)$, $\underline{\underline{E}}(1, 1)$ respectively. We shall discuss the way of implementing a relation of the type $\underline{a}_x \underline{Z}(k, \ell) \underline{b}_x$, in general, in Sections 3 and 4. However, we may mention here that the implication relation in QL-1A takes the form

$$\underline{a}_x \underline{I} \underline{b}_x \equiv \underline{a}_x^n \underline{Q} \underline{b}_x, \quad \underline{a}_x^n = \underline{a}_x \underline{N} \quad (10)$$

In essence, in QL-1A, we implement $\underline{a}_x \underline{Z} \underline{b}_x$ by performing the logical operation demanded by the relation $\underline{a}_x \underline{Z} \underline{b}_x = \underline{c}_x$, for each $x = 1$ to n , /and then finding out the QBA state of $(\underline{c}_1, \underline{c}_2, \dots, \underline{c}_n)$ by referring back to Table 1. However, it should be noted that

the quantifier states of \underline{ax} , \underline{bx} , \underline{cx} in (9a,b,c) are those describable by $q(1)$ to $q(8)$ of the BA-3 algebra of QBA listed in Table 1, although the relations on the r.h.s of (9a,b,c) are between the individual SNS terms of the $(2,n)$ -vector corresponding to the /quantified states. Because of this, the quantified terms themselves can be combined by using the Boolean operators \underline{U} and \underline{V} of QL-2. As mentioned in MR-53, these can be represented by the symbols of Boolean sum (\oplus) and Boolean product (\otimes), applied to $q(1)$ to $q(8)$. In their application to QL-1A relations, it can be verified that the QL-1A operator \underline{Z} and the QL-2 operators \oplus and \otimes have the distributive property listed in (11):

$$(\underline{ax} \oplus \underline{bx}) \underline{Z} \underline{cx} = (\underline{ax} \underline{Z} \underline{cx}) \oplus (\underline{bx} \underline{Z} \underline{cx}), \underline{Z} = \underline{A}, \underline{O}(\underline{I}), \underline{E} \quad (11a)$$

$$(\underline{ax} \otimes \underline{bx}) \underline{Z} \underline{cx} = (\underline{ax} \underline{Z} \underline{cx}) \otimes (\underline{bx} \underline{Z} \underline{cx}), \underline{Z} = \underline{A}, \underline{O}(\underline{I}), \underline{E} \quad (11b)$$

We shall now proceed to give some of the details regarding the properties of the matrix connectives \underline{Z} in QL-1A.

(b) Outline of the theory of QL-1A matrix connectives

We utilize the definition of the three operators \underline{A} , \underline{O} , \underline{E} as given by (8a,b) and (9a,b,c), and that of the eight quantifier states of QBA as given in column 4 of Table 1. Then, the 3×3

multiplication tables for these three operators, along with that of \underline{I} as defined by Eq.(10), for the three generators $\underline{V}, \underline{\Sigma}, \underline{\Phi}$ of QBA, can be obtained. These are shown in Tables 2(a) to 2(d) (the details are omitted). The Boolean operator c for complementation is employed in the form of weak negation (\underline{M}), and is defined in the same manner as for QL-2, in (12):

$$\underline{a} \underline{M} = \underline{b} : \underline{a}^c 1 = \underline{b} 1, \dots, \underline{a}^c N = \underline{b} N ; (\underline{a}^c = \underline{b}) \equiv (q(\underline{1}^c ax) = q(\underline{1}bx)) \quad (12)$$

The strong negation (\underline{N}) is also defined in the same manner as the matrix connective for QL-2, and takes the form of Eq.(13):

$$\underline{a} \underline{N} = \underline{b} : \underline{a} 1 \underline{N} = \underline{b} 1, \dots, \underline{a} N \underline{N} = \underline{b} N ; (\underline{a}^N = \underline{b}) \equiv (q(\underline{1}^N ax) = q(\underline{1}bx)) \quad (13)$$

Using these, the De Morgan relations and other inter-relationships between the operators $\underline{A}, \underline{O}, \underline{I}, \underline{E}$ are given in (14a to e).

De-Morgan relations for QL-1A

$$\underline{N} \underline{A} \underline{N} = \underline{O}^c \equiv \underline{a}^N x \underline{A} \underline{b}^N x = (\underline{ax} \underline{O} \underline{bx})^N = \underline{c}^N x, \text{ with } \underline{c} \text{ of (9a) (14a)}$$

$$\underline{N} \underline{O} \underline{N} = \underline{A}^c \equiv \underline{a}^N x \underline{O} \underline{b}^N x = (\underline{ax} \underline{A} \underline{bx})^N = \underline{c}^N x, \text{ with } \underline{c} \text{ of (9b) (14b)}$$

Relation between implication and disjunction in QL-1A

$$\underline{N} \underline{O} = \underline{I} \equiv \underline{a}^N x \underline{O} \underline{bx} = \underline{ax} \underline{I} \underline{bx} \quad (14c)$$

$$\underline{A} \underline{N} = \underline{I}^c \equiv \underline{ax} \underline{A} \underline{b}^N x = (\underline{ax} \underline{I} \underline{bx})^N \quad (14d)$$

Table 2. Multiplication tables for the/matrix connectives
A, O, I, E ^{*} for the generators of QBA

(a) AND (A)

<u>A</u>	V	Σ	Φ
V	V	Σ	Φ
Σ	Σ	\wedge	Φ
Φ	Φ	Φ	Φ

(b) OR (O)

<u>O</u>	V	Σ	Φ
V	V	V	V
Σ	V	E	Σ
Φ	V	Σ	Φ

(c) IMPLY (I)

<u>I</u>	V	Σ	Φ
V	V	Σ	Φ
Σ	V	E	Σ
Φ	V	V	V

(d) EQUIVALENT (E)

<u>E</u>	V	Σ	Φ
V	V	Σ	Φ
Σ	Σ	Δ	Σ
Φ	Φ	Σ	V

*These four connectives are given the logical symbols, respectively, with an extra wavy line below the corresponding symbols in sentential logic.

+The tables give $c'x$ corresponding to $a'x \wedge b'x = c'x$

Contrapositive implication relation

$$\underline{\sim} \underline{\sim} \underline{\sim} N = I^t \equiv a^n x \text{ I } b^n x = \underline{\sim} x \text{ I } \underline{\sim} ax \quad (14e)$$

As illustrations, we give the following examples:

$$(14a) : (\forall x)(\neg \underline{\sim} ax) \wedge (\exists x)(\neg \underline{\sim} bx) \equiv \neg ((\forall x)(\underline{\sim} ax) \vee (\exists x)(\underline{\sim} bx))$$

$$(14b) : (\exists x)(\underline{\sim} ax) \wedge (\exists x)(\neg \underline{\sim} bx) \equiv \neg ((\exists x)(\underline{\sim} ax) \Rightarrow (\exists x)(\underline{\sim} bx))$$

$$(14e) : (\forall x)(\neg \underline{\sim} ax) \Rightarrow (\forall x)(\neg \underline{\sim} bx) \equiv ((\forall x)(\underline{\sim} bx) \Rightarrow (\forall x)(\underline{\sim} ax))$$

where the QL-1 operator corresponding to the SNS operators of PC are differentiated by the wavy line below their symbols as in $\underline{\sim}$, $\underline{\vee}$, $\underline{=}$, $\underline{\equiv}$, corresponding to \underline{A} , \underline{Q} , \underline{I} , \underline{E} respectively.

From Tables 2(a) to 2(d) for the generator states, we can construct the corresponding 8x8 truth tables for all QBA states, by the simple fact that any QBA state is a Boolean sum of atmost three generator states. We have only to use the distributive law between Boolean sum and the operators \underline{A} , \underline{Q} , \underline{I} and \underline{E} , of the form shown in (15a,b).

$$(q_1 \oplus q_2) \underline{A} (q_3 \oplus q_4) = q_1 \underline{A} q_3 \oplus q_1 \underline{A} q_4 \oplus q_2 \underline{A} q_3 \oplus q_2 \underline{A} q_4 \quad (15a)$$

$$(q_1 \oplus q_2) \underline{Q} (q_3 \oplus q_4) = q_1 \underline{Q} q_3 \oplus q_1 \underline{Q} q_4 \oplus q_2 \underline{Q} q_3 \oplus q_2 \underline{Q} q_4 \quad (15b)$$

The tables so obtained are given in the next pages as
Tables 3(a) to 3(d).

It will be seen from Table 3(a) that the four standard quantifiers $\forall, \exists, \wedge, \Phi$ yield only one of the same four states as a result of conjunction \underline{A} , except that $\exists \underline{A} \exists = \Delta$, the indefinite state. A set-theoretical explanation of the occurrence of Δ has been given in IS-1, for the operator "into" (\otimes_L) of QL-1, which is really equivalent to the new QL-1A operator \underline{A} , for the range of applicability of \otimes_L — namely (\forall, \exists) .

In Tables 3(b),(c), (d) also, it is found that the 4x4 multiplication tables for the four standard quantifier states $\forall, \exists, \wedge, \Phi$, yield only again one of the four states, except for the additional indefinite state Δ . On putting back Δ also as an input, once again it is seen that no further new states are introduced. So also, on introducing the contradictory state ϕ , no new state other than this is introduced.

Therefore, Tables 3(a) to 3(d) can be said to indicate that QL-1A calculus using only the matrix operators \underline{A} , \underline{Q} , \underline{I} and \underline{E}

in QL-1A

Table 3. Full multiplication tables/for the matrix
connectives $\Lambda, \exists, \wedge, \Phi$ for the eight states of QBA

(a) AND^1

Λ	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
\forall	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
\exists	\exists	Δ	\wedge	Φ	\wedge	Δ	Δ	ϕ
\wedge	\wedge	\wedge	\wedge	Φ	\wedge	\wedge	\wedge	ϕ
Φ	Φ	Φ	Φ	Φ	Φ	Φ	Φ	ϕ
Σ	Σ	\wedge	\wedge	Φ	\wedge	\wedge	\wedge	ϕ
Δ	Δ	Δ	\wedge	Φ	\wedge	Δ	Δ	ϕ
Θ	Θ	Δ	\wedge	Φ	\wedge	Δ	Θ	ϕ
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

* The four entries in the top l.h. corner of the table for the standard quantifiers \forall, \exists are the same as those for the operator \otimes_L which was described in IS-1 using lattice algebra for QL-1.

will be complete, if we have only the six states ($\forall, \exists, \wedge, \phi, \Delta, \emptyset$) of QBA. However, as will be shown later, true Boolean connectives applied to the QBA states (which will follow the rules of QL-2), can yield the other two possible QBA states, ξ and Θ . Therefore, these have also been included in Tables 3(a) to 3(d). Then, it is observed that the multiplication tables are once again complete, because the matrix product always leads back to one of the eight states, with the proviso that ξ , or Θ , arises only if at least one of the components in the relation $\underline{a} \underline{Z} \underline{b}$ is ξ , or Θ , respectively. We shall comment further on the properties of these connectives when we come to applications.

(c) Implementation of QL-1A unary and binary relations

(i) Binary relation in QL-1A

This has the following structure :

$$\underline{a}'x, \underline{b}'x, \underline{a}x \underline{Z} \underline{b}x = \underline{c}x \mapsto \underline{c}'x \quad (15a)$$

where $\underline{a}x$ and $\underline{b}x$ are dummy symbols and $\underline{a}'x$ and $\underline{b}'x$ are input quantified terms which lead to the quantified output term $\underline{c}'x$ as in (15b):

$$\underline{a}'x \underline{Z} \underline{b}'x = \underline{c}'x, \quad \underline{Z} = A, O, I, E. \quad (15b)$$

Eq.(15b) utilizes the data in Tables 2(a) to (d) for $\underline{z} = \underline{A}(1, 1)$, $\underline{Q}(1, 1)$, $\underline{I}(1, 1)$, $\underline{E}(1, 1)$ respectively. For the more general case of (15a) in which $\underline{z} = \underline{z}(k, \ell)$, the implementation can be carried out in terms of (15b) as follows. First, $\underline{a}''x$, $\underline{b}''x$ are calculated as in (16a) and (16b):

$$\underline{a}''x = \underline{a}'x \text{ or } \underline{a}'^nx, \text{ according as } k = 1, 2 \quad (16a)$$

$$\underline{b}''x = \underline{b}'x \text{ or } \underline{b}'^nx, \text{ according as } \ell = 1, 2 \quad (16b)$$

Then the output $\underline{c}'x$ is given by (16c):

$$\underline{a}'x \underline{z}(k, \ell) \underline{b}'x \equiv (\underline{a}''x \underline{z}(1, 1) \underline{b}''x) = \underline{c}'x \quad (16c)$$

The implementation of Eq.(16c) requires only the four tables 2(a) to (d) and the calculation for $\underline{z}(k, \ell)$ is reduced to that for $\underline{z}(1, 1) \equiv \underline{z}$.

(ii) Unary relation in QL-1A

As already mentioned, the relations in QL-1A arrives during the implementation of QL-1 statements for given inputs. Thus, Eq.(15a) arises from the QL-1 equation

$$\underline{a}'x, \underline{b}'x, (\forall x)(\underline{a}x \underline{z} \underline{b}x = \underline{c}x) \mapsto \underline{c}'x \quad (17a)$$

In a very similar manner, the unary QL-1 equation

$$\underline{a}'x, (\forall x)(\underline{a}x \underline{z} = \underline{b}x) \mapsto \underline{b}'x \quad (17b)$$

leads to the QL-1A unary relation

$$\underline{a}'x, \underline{a}x \underline{Z} = \underline{b}x \mapsto \underline{b}'x \quad (18)$$

In this, as in (15a), $\underline{a}x$ is a dummy symbol and the input quantified term is $\underline{a}'x$, and the operation of the QL-1 connective \underline{Z} leading to the output quantified term $\underline{b}'x$ has the same significance as for the binary relation. Therefore, if $\underline{a}'x$ has the full description $q(\underline{ia}'x)(\underline{a}'x)$, then, the SNS operator \underline{Z} is applied to each of the components of the $(2,N)$ -vector representing $\underline{a}'x$ to obtain the components of the $(2,N)$ -vector representing $\underline{b}'x$. Therefore, the equations for the implementation of (18) are as follows:

$$\text{Let } \underline{a}'x = q(\underline{ia}'x)(\underline{a}'x) \quad (19a)$$

$$\text{Calculate } \underline{a}'x \underline{Z} = \underline{b}'x \quad (19b)$$

$$\text{Then } \underline{b}'x = q(\underline{ia}'x)(\underline{b}'x) \quad (19c)$$

The equations (19a,b,c) give the implementation of (18) for $\underline{Z}(1, 1)$. For the more general case where $\underline{Z} = \underline{Z}(k, \ell)$, a similar set of equations can be obtained and are given in (20). First calculate $\underline{a}''x$ from (20a):

$$\underline{a}''x = \underline{a}'x \text{ or } \underline{a}'^n x, \text{ according as } k = 1, 2 \quad (20a)$$

Then

$${}^{\prime}x \underline{\underline{z}} = \underline{\underline{b}}{}^{\prime}x ; \quad \underline{\underline{b}}{}^{\prime}x = q(\underline{\underline{a}}{}^{\prime}x)(\underline{\underline{b}}{}^{\prime}x) \quad (20b,c)$$

Then the output $\underline{\underline{b}}{}^{\prime}x$ is given by

$$\underline{\underline{b}}{}^{\prime}x = \underline{\underline{b}}{}^{\prime}x \text{ or } \underline{\underline{b}}{}^{\prime n}x, \text{ according as } \ell = 1, 2 \quad (20d)$$

The above theory of QL-1A binary and unary relations will be applied immediately in the next section for the implementation of the corresponding elementary relation in QL-1.

3. Implementation of the QL-1 binary relation

The standard form of the binary relation and its implementation in QL-1 have been indicated in Eq.(4). It was also mentioned in that connection that the implementation requires the definition and formulae pertaining to the QL-1 matrix connectives $\underline{\underline{A}}, \underline{\underline{Q}}, \underline{\underline{E}}$. Since these connectives and their multiplication tables have been listed in Section 2, the formulae for the implementation of the binary relation can be written very simply. Thus, the binary relational equation is

$$q(\underline{\underline{a}}{}^{\prime}x), q(\underline{\underline{b}}{}^{\prime}x), q(\underline{\underline{z}}x)(\underline{\underline{a}}x \underline{\underline{z}}(k, \ell) \underline{\underline{b}}x = \underline{\underline{c}}x) \mapsto q(\underline{\underline{c}}{}^{\prime}x), \text{ or} \\ \underline{\underline{t}}(\underline{\underline{z}} \mid \underline{\underline{a}}', \underline{\underline{b}}') \quad (21a,b)$$

In implementing these, we first implement the QL-1A binary relation of (21c) :

$$q(\underline{ia}'x), q(\underline{ib}'x), \underline{ax} \underline{Z}(k, \ell) \underline{bx} \mapsto q(\underline{ic}''x) \quad (21c)$$

The steps involved in this implementation are

$$q(\underline{ia}''x) = q(\underline{ia}'x), q(\underline{i}^n a''x), \text{ for } k = 1, 2 \text{ respectively} \quad (22a)$$

$$q(\underline{ib}''x) = q(\underline{ib}'x), q(\underline{i}^n b''x), \text{ for } \ell = 1, 2 \text{ respectively} \quad (22b)$$

$$q(\underline{ia}''x) \underline{Z} q(\underline{ib}''x) = q(\underline{ic}''x), \underline{Z} = \underline{A}, \underline{Q} \text{ or } \underline{E} \text{ according as} \\ \underline{Z} = \underline{A}, \underline{Q}, \underline{E} \quad (\underline{I}(k, \ell) = \underline{Q}(k^n, \ell)) \quad (23)$$

The outputs in (21a,b) can be obtained from $q(\underline{ic}''x)$ as in (24a,b)

$$q(\underline{ic}''x) \underline{A} q(\underline{iz}x) = q(\underline{ic}'x) \quad (24a)$$

$$\underline{t}(q(\underline{iz}x) | q(\underline{ic}''x)) = \underline{t}(\underline{Z} | \underline{a}', \underline{b}') \quad (24b)$$

Of these two formulae, (24b) is well founded, and can always be used. We are not yet quite sure of the correct connective to be used in (24a) and this will be clarified after discussing the QL-1 unary relation in the next section.

Thus, for the input quantifier states $\underline{a}'x$ and $\underline{b}'x$, the output of the binary relation can be obtained, either as a quantifier state $\underline{c}'x$, or as a truth value $\underline{t}(\underline{Z} | \underline{a}', \underline{b}')$.

The latter is probably more useful, as it gives at a glance, the information whether the QL-1 relation $(\underline{q}_Z x)(\underline{a}x \underline{Z} \underline{b}x)$ is completely satisfied by the inputs $\underline{a}'x$ and $\underline{b}'x$, or whether the negation of this relation is satisfied, or whether both are satisfied, according as the truth value is T, F or D respectively. On the other hand, if the question is asked as to the nature of the quantifier, for which the relation formed by the predicate is satisfied, for the given quantifier states of the two inputs, then this can also be answered by the quantifier state denoted by $\underline{c}'x$. We shall illustrate these by two examples.

Problem 1

The data given are

$$\exists(\underline{a}'), \quad \Phi(\underline{b}'), \quad (\forall x)(\underline{a}x \underline{A}(1,2) \underline{b}x) \quad (25a)$$

Then,

$$\underline{a}''x = \exists, \quad \underline{b}''x = \Phi^n = \forall, \quad \exists \underline{A} \forall = \exists = \underline{c}''x \quad (25b)$$

$$\underline{t}(\underline{Z} \mid \underline{a}', \underline{b}') = \underline{t}(\forall \mid \exists) = D \quad (25c)$$

Problem 2

The input data are

$$\exists(\underline{a}'), \quad \exists(\underline{b}'), \quad (\Phi x)(\underline{a}x \underline{Q}(1,2) \underline{b}x) \quad (26a)$$

$$\underline{a}''x = \exists, \underline{b}''x = \exists^n = \exists, \exists 0 \exists = \exists = c''x \quad (26b)$$

$$\underline{t}(\underline{Z} | \underline{a}', \underline{b}') = \underline{t}(\underline{\Phi} | \exists) = F \quad (26c)$$

It will be noticed that the above algorithm, in (22) and (23) for the QL-1 binary relation, is very much shorter than the corresponding one which was proposed in IS-1. This has been possible because all the calculations involving the 3-vectors denoting quantifier states, and the application of the binary relations \underline{A} , \underline{Q} or \underline{E} , are contained in the relevant tables as given in the previous section. It is suggested that the 3x3 tables, Tables 2(a), (b), (d) only need to be stored for these three connectives \underline{A} , \underline{Q} , \underline{E} respectively, corresponding to the generator states, and that, for a matrix relation involving mixed states, they be written as Boolean sums of their generator states, and the resultant worked out as indicated below.

Denoting the generator states $q(1), q(3), q(5)$ by $q(g_1)$ and the QL-1 operators $\underline{A}, \underline{Q}, \underline{E}$ by \underline{Z} , the matrix product $q(\underline{i}) \underline{Z} q(\underline{j})$, of the mixed states $q(\underline{i})$ and $q(\underline{j})$ can be obtained as follows:

$$\text{Suppose } q(\underline{i}) = \bigoplus_i q(g_i) , \quad q(\underline{j}) = \bigoplus_j q(g_j) , \quad (27a)$$

then

$$q(\underline{i}) \preceq q(\underline{j}) = \bigoplus_i \bigoplus_j (q(g_i) \preceq q(g_j)) \quad (27b)$$

This is the only extra formula that is needed for implementing Eqs. (21) and (22), since the formula for the relative truth value $\underline{t}(\underline{a} | \underline{a}')$ is the same as in QL-2, since we are using the QL-2 notation for the quantifier states.

4. Implementation of the QL-1 unary relation

This problem looks deceptively simple, but gives a lot of difficulty in its formulation via the QBA states of QL-2 for quantifiers. Therefore, we shall give a brief recapitulation of the corresponding formulae using the lattice algebra symbolism for quantifier states, which was developed in ALOG-48 and IS-1 for QL-1 logical relations.

(a) Formulae for the standard quantifiers $\forall, \exists, \wedge, \exists$ (IS-1 formulation via lattice algebra)

In IS-1, we had given a very simple formulation using a special designation of QL-1 states by lattice numbers for the implementation of the QL-1 relation (as given in Eq.(3))

for any of the four standard quantifiers mentioned above.

We shall first state this and then indicate how those formulae can be restated in terms of QL-2 states, as we have developed for this report. Thus, using the lattice number $q = 1, 2$, and the SNS state number $k = 1, 2$, a QL-1 quantifier \underline{q} is denoted by (q, k) . The unary relation (3) can be restated in this form as

$$(q_a, k_a), q(\underline{izx}) (\underline{ax} \underline{z} = \underline{bx}) \mapsto (q_b, k_b) \quad (28a)$$

where

$$q(\underline{izx}) = (q_z, 1), \quad q_z = 2 \text{ or } 1, \text{ (corresponding to } \forall, \exists) \quad (28b)$$

An unstated assumption is made that the relation can have only a quantifier state $q(\underline{izx})$ corresponding to the SNS state of the predicate being T, or $k_z = 1$. Then, the implementation of (28a) leads to the following:

$$q_a, \bigoplus_L q_z = q_b; \quad s(k_a) \underline{z}(k, \ell) = s(k_b); \quad \underline{b}' = (q_b, k_b) \quad (28c)$$

In the implementation of the formulae in (28c), the SNS matrix multiplication is in the standard EVMF formulation and the lattice product involving the operator "into" can be stated as in (28d) :

$$\forall = \forall \quad \exists = \exists \quad \forall = \exists, \quad \exists \bigoplus_L \exists = \Delta \quad (28d)$$

(b) Formulation for standard quantifier states using matrix algebra and QL-1 connectives \underline{Z} .

We shall first generalize this formulation to the cases where the quantifier \underline{q}_Z of the relation, as well as the quantifier \underline{q}_a of the input, can be any one of the four states $\forall, \exists, \wedge, \Phi$. In such a case, we first convert the statement containing the quantifiers \wedge and Φ into a form involving only \exists and \forall . Thus,

$$(\Phi x)(\underline{a}x \underline{Z} \underline{b}x) \equiv (\forall x)(\underline{a}x \underline{Z}^c \underline{b}x) \quad (29a)$$

$$(\wedge x)(\underline{a}x \underline{Z} \underline{b}x) \equiv (\exists x)(\underline{a}x \underline{Z}^c \underline{b}x) \quad (29b)$$

Then, the theory for any of the four standard quantifier states for the input $\underline{a}'x$ can be formulated following the line of argument given below. We first convert the QL-1 unary relation

$$\underline{a}'x, (\underline{q}_Z x)(\underline{a}x \underline{Z}(k, \ell) = \underline{b}x) \mapsto \underline{b}'x \quad (30a)$$

into a formula containing a QL-1A relation by inputting $\underline{a}'x$ "into" the relation within the brackets in (30a). This consists of two parts. First we obtain the effective quantifier state of the input $\underline{a}'x$ for the QL-1A relation by using the operator "into" which has been defined in IS-1 as

in (30b):

$$a'x \quad \forall_L q_z = \underline{a}'_1x = (\underline{q}_{a'_1}x) (\underline{a}'_1x) \quad (30b)$$

Then the predicate of the QL-1 relation (30a) is converted into a QL-1A relation with input \underline{a}'_1x , in the same format as the standard QL-1A unary relation in Eq.(18), as follows:

$$\underline{a}'_1x, \quad \underline{a}x \underline{z}(k, \ell) = \underline{b}x \mapsto \underline{b}'x \quad (30c)$$

The solution of (30c), with the input in (30b) for \underline{a}'_1x , follows from Eq. (19a,b,c) as

$$\underline{a}'_1x \underline{z}(k, \ell) = \underline{b}'x \quad (30d)$$

$$(\underline{q}_{a'_1}x) (\underline{b}'x) = \underline{b}'x \quad (30e)$$

As will be readily seen, Eqs. (30(d) and (e) are workable using the algebra already developed so far. The only equation that requires special considerations is (30b), in which the operator \bigotimes_L occurs. We shall give below some considerations which indicate that this operator has many of the properties of the QL-1A operator \underline{A} .

Thus, for the two common quantifier states \forall and \exists the 2x2 multiplication table for \bigotimes_L given in (28d) is identical with the corresponding table for \underline{A} shown in Table 3(a) for AND.

However, the similarity ends there. When the other two quantifier states Φ and Λ are input, we convert them into $V(F)$ and $\exists(F)$ respectively, and the resultant input for the QL-1A relation, in these cases, are given by

$$\begin{aligned} V(F) \otimes_L V &= V(F) , & V(F) \otimes_L \exists &= \exists(F) , \\ \exists(F) \otimes_L V &= \exists(F) , & \exists(F) \otimes_L \exists &= \Delta(F) \quad \Lambda \end{aligned} \quad (31)$$

In obtaining the equations in (31), we follow the results for the operation of QL-1 AND ($\underline{\Lambda}$) between the two states V, \exists occurring for q_a , and q_z , and the SNS "and" ($\underline{\Lambda}$) between the terms \underline{a}' and $s_z = s(1) = T$. The four equations in (31), along with the four in (28d), are all that are necessary for dealing with the classical quantifier states $V, \exists, \Lambda, \Phi$ for this problem. The effective 4x2 table for \otimes_L is given below. It will be noticed that it only uses the 2x2 multiplication table for the QL-1A operator $\underline{\Lambda}$, which is marked out in Table 2(a).

Table 4. Multiplication table for the operator "into" in

$$\underline{a}'x \otimes_L \underline{q}_z x = \underline{a}'_1 x$$

\underline{a}'	\underline{q}_z	V	\exists
$V = V(T)$		$V(T)$	$\exists(T)$
$\exists = \exists(T)$		$\exists(T)$	Δ
$\Lambda = \exists(F)$		$\exists(F)$	Δ
$\Phi = V(F)$		$V(F)$	$\exists(F)$

Therefore, the QBA formalism for QL-1 unary implementation takes the following form:

Denote $q(\underline{1})$ by a star for $\underline{1} = \underline{2}$ and $\underline{5}$, corresponding to \wedge and $\bar{\phi}$ respectively. The unary relation (30a) is stated as follows:

$$q(\underline{1a}'x), \quad q(\underline{1Z}x)(\underline{ax} \underline{Z}(k, \ell) = \underline{bx}) \mapsto q(\underline{1b}'x) \quad (31a)$$

Replace the relation by

$$q(\underline{1Z}'x) (\underline{ax} \underline{Z}'(k, \ell) = \underline{bx}) \quad (31b)$$

where

$$\underline{1Z}' = \underline{1Z}, \quad \underline{Z}' = \underline{Z} \quad \text{if } q(\underline{1Z}x) \text{ is unstarred, and} \quad (31c)$$

$$\underline{1Z}' = \underline{1}^n \underline{Z}, \quad \underline{Z}' = \underline{Z}^c \quad \text{if } q(\underline{1Z}x) \text{ is starred} \quad (31d)$$

Similarly, replace

$$q(\underline{1a}'x) \text{ by } q(\underline{1a}''x)(\underline{a}''x) \quad (31e)$$

where

$$\underline{1a}''x = \underline{1a}'x \quad \text{and} \quad \underline{a}''x = T \quad \text{if it is unstarred, and} \quad (31f)$$

$$\underline{1a}''x = \underline{1}^n \underline{a}'x \quad \text{and} \quad \underline{a}''x = F \quad \text{if it is starred} \quad (31g)$$

Then,

$$q(\underline{1a}''x) \wedge q(\underline{1Z}'x) = q(\underline{1b}''x) ; \quad \underline{a}'' \underline{Z}'(k, \ell) = \underline{b}'' \quad (31h)$$

$$\underline{b}' \quad q(\underline{1b}'x) = q(\underline{1b}''x)(\underline{b}''x) \quad (31k)$$

As will be clear from Table 4, the output $\underline{b}'x$ can have any one of the four standard quantifier states, and also have, in addition, the indefinite state Δ . We have purposely omitted the designation of the SNS truth value attached to the symbol Δ . This is because $\Delta(T) \equiv \Delta(F) \equiv \Delta(D)$. For the purpose of the check for contradiction given in the next para, $\Delta(D)$ is to be used.

It can happen that \underline{b}'' in (31k) is equal to X, indicating the contradiction $\underline{a}'' \vee s(k) = X$, when the corresponding vidya check between the input quantifier state $q(\underline{1a}'x)$, and the quantifier $q(\underline{1Z}'x)$ of the relation, should be made. ^{be found that} It will / this check also leads to the impossible quantifier state ϕ , showing that the origin of the contradiction is to be ascribed to that between the input term, and the quantifier state of the binary relation employed.

This happens, for instance, with the input $(\underline{\phi} x)(\underline{a}'x)$ into the relation $(\underline{\exists} x)(\underline{ax} \underline{A}(1,1) = \underline{bx})$. Since $(\underline{\phi} x)$ is a starred state, it is replaced by $(\underline{\forall} x)(\underline{a}''x = F)$ using (31g). Then we obtain,

$$(\underline{\forall} x) \underline{A} (\underline{\exists} x) = (\underline{\exists} x) = q(\underline{1b}''x) ; F \underline{A}(1, 1) = X = \underline{b}''x ,$$

indicating contradiction (32a)

It is readily seen that the contradiction in (32a) arises from $\underline{a}'' \underline{V} s(k)$ in the form

$$\bar{F} \underline{V} s(1) = F \otimes T = X \quad (32b)$$

which shows that the contradiction is due to the input to being F, the predicate/while the SNS connective is $\underline{A}(1, 1)$ standing for $\underline{a} \wedge \underline{b}$, with which it is incompatible. Also, on applying the QL-2 vidya check \underline{V} , as mentioned above, between the quantifier states, of the input and the relation, we readily verify that

$$\underline{\exists} \underline{V} \underline{\exists} = (0 \ 0 \ 1) \otimes (1 \ 1 \ 0) = (0 \ 0 \ 0) \quad (32c)$$

Thus, we have obtained the theory of the unary relation in QL-1 for the four standard quantifier states. We shall consider the extension of this theory to the four other QBA states $\underline{\Sigma}, \underline{\Delta}, \underline{\Theta}, \underline{\phi}$ in the next section, after giving a short account of the QL-1 binary reverse relation in the next sub-section.

(c) QL-1 binary reverse relation leading to unary relation

From the binary forward relation (21), the binary reverse relation can be defined as follows:

$$q(\underline{1}Zx) (\underline{a}x \underline{Z}(k, \ell) \underline{b}x) = q(\underline{1}c'x) \quad (33a)$$

In (33a), the binary relation on the l.h.s has $q(\underline{1}Zx) = \forall$ or \exists , and has $k_Z = 1$, so that \underline{q}_Z has only the values (1, 1) and (2, 1). Similarly $q(\underline{1}c'x)$ when written in QL-1 notation, in terms of these two quantifiers \forall, \exists , is $(q_{c'}, k_{c'})$, where $q_{c'} = 1$ or 2 and $k_{c'} = 1$ or 2. Then, Eq.(33a) becomes equivalent to the unary relations (33b,c) :

$$q(\underline{1}Z'x) (\underline{a}x \underline{Z}'(k, \ell) = \underline{b}x) ; q(\underline{1}Z'x) (\underline{b}x \underline{Z}'^t(k, \ell) = \underline{a}x) \quad (33b,c)$$

where

$$\underline{q}_Z \wedge \underline{q}_{c'} = \underline{q}_{Z'} ; \underline{Z}' = \underline{Z}, \underline{Z}^c, \text{ according as } k_{c'} = 1, 2 \text{ respectively} \quad (33d,e)$$

The proof of these follows from that given in IS-1, by extending its range from \forall, \exists , to all the four standard quantifier states.

5. QBA implementation of QL-1 unary relations(a) Statement of the problem

This section reads continuously from the end of Section 3 (page 23). The unary relational equation in QL-1 takes the following form:

$$q(\underline{ia}'x), \quad q(\underline{izx})(\underline{ax} \underline{Z}(k, \ell) = \underline{bx}) \mapsto q(\underline{jb}'x) \quad (34)$$

This can be rewritten in the form of a binary forward QL-1 relation as in (35):

$$\underline{a}'x \underline{Z}(k, \ell) \underline{b}'x = \underline{c}'x \equiv q(\underline{izx}) \quad (35)$$

In fact, using the notation for a binary reverse relation, Eq.(35) can be recasted in the form equivalent to (34) as follows:

$$\underline{a}'x \quad (\underline{Z}(k, \ell), \underline{c}'x) = \underline{b}'x ; \quad \underline{c}'x \equiv q(\underline{izx}) \quad (36)$$

Essentially, in Eq.(34), we are given the SNS unary relation within the bracket quantified by $q(\underline{izx})$ to be true, and we are given the quantified term $q(\underline{ia}'x)$ and ^{asked} to find the corresponding quantified term $q(\underline{jb}'x)$ which would make the relation true. In other words, what is a unary forward relation

is to be understood as arising from a QL-1 binary reverse relation from the forward binary relation given in (35). This is in the spirit of the definition of unary relations, in general, in BVMF, as arising from reversing a binary relation, given the state of the output term, and seeking to find out one of the input terms given the other. In fact, Eq.(36) expresses this explicitly by saying that we are given the input $\underline{a}'x$ and the operator $\underline{Z}(k, \ell)$ which, with $\underline{a}'x$ and $\underline{b}'x$, would give the final output $\underline{c}'x$, corresponding to the quantifier state $q(\underline{i}2x)$.

The great merit of this way of formulating the problem is that we can straightaway utilize the QL-1A theory in Section 2(b) for the forward relation, and work out, from the multiplication tables in Tables (2) and (3), the output $\underline{b}'x$, by deriving it from the input $\underline{a}'x$ and the resultant $\underline{c}'x$. We shall illustrate this procedure in the next subsection and work out, in particular, the 3×3 ^{tables} / for the operators $\underline{Z} = \underline{A}(1, 1), \underline{O}(1, 1), \underline{I}(1, 1)$ and $\underline{E}(1, 1)$. Thereafter, we shall work out the full 8×8 table for all the quantifier states in QBA and then indicate how the theory can be generalized to $\underline{Z}(k, \ell)$ from $\underline{Z}(1, 1)$.

(b) QL-1A binary reverse relation for the generators of QBA

The great advantage of using the generators as the basis for describing all possible quantifier states is that they form a set of orthogonal vectors in BA-3, and the reversal of the results in Table 2(a-d) for the four matrix connectives \underline{A} , \underline{O} , \underline{I} , \underline{E} can be done merely by inspecting these tables. The resultant tables for the binary reverse operators $\overleftarrow{\underline{A}}$, $\overleftarrow{\underline{O}}$, $\overleftarrow{\underline{I}}$, $\overleftarrow{\underline{E}}$ are given in Table 5. We shall describe in detail the working out of these tables for \underline{A} and \underline{E} .

Considering row 1 of Table 5(a) first, the input $\underline{a}'x$ is \forall , and we are required to find out what will be the quantifier state of $\underline{b}'x$ for which the output $\underline{c}'x$ will be \forall , \exists , $\bar{\Phi}$, respectively. This creates no difficulty because each \underline{c}' is produced only by one \underline{b}' in Table 2(a) and the corresponding entries are the same in Table 5(a) also. In the same way, considering the third row of Table 2(a), it contains throughout the entry $\bar{\Phi}$ only for all three columns, in the body of the table corresponding to this row. Hence in Table 5(a), it is only possible to have $\underline{c}' = \bar{\Phi}$, and \underline{c}' cannot be \forall or \exists , so that we have the entries ϕ , ϕ , Δ — i.e. neither $\underline{b}' = \forall$ nor $\underline{b}' = \exists$ can give rise to $\underline{c}' = \bar{\Phi}$; and any value, namely

Table 5. Output of the QL-1 unary relation* for the generator states of QBA as inputs for all four types of matrix connectives

(a) $\underline{A}(1, 1)$

$\underline{a}' \backslash \underline{c}'$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\Phi}$	$\underline{\exists}$	$\underline{\wedge}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Delta}$

(b) $\underline{O}(1, 1)$

$\underline{a}' \backslash \underline{c}'$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$
$\underline{\Phi}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$

(c) $\underline{I}(1, 1)$

$\underline{a}' \backslash \underline{c}'$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$
$\underline{\Phi}$	$\underline{\Delta}$	$\underline{\Phi}$	$\underline{\Phi}$

(d) $\underline{E}(1, 1)$

$\underline{a}' \backslash \underline{c}'$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\Sigma}$	$\underline{\wedge}$	$\underline{\Sigma}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Sigma}$	\underline{V}

* In the relation, $\underline{a}'x, q(\underline{iZx})(\underline{ax} \underline{Z} = \underline{bx}) \mapsto \underline{b}'x$, we represent $q(\underline{iZx})$ by $\underline{c}'x$ and the tables list the output $\underline{b}'x$ corresponding to the inputs $(\underline{a}'x, \underline{c}'x)$ for $\underline{Z} = A, O, I, E$

\vee , ξ or Φ for \underline{b}' will lead to $\underline{c}' = \Phi$ corresponding to
 $\underline{a} = \Phi$.

The middle row is however not so simple to work out.

Considering first $\underline{a}' = \xi$, $\underline{c}' = \vee$, it is clear that

Table 2(a) does not contain the generator state \vee in any of the entries in the middle row. Therefore, there is no solution to this combination in the binary reverse relation and this is indicated by the null set in row 2, column 1, ^{of} Table 5(a)

Taking ^{next}, the combination $\underline{a}' = \xi$, $\underline{c}' = \xi$, the value of $\underline{c}' = \xi$ occurs corresponding to both $\underline{b}' = \vee$ and $\underline{b}' = \xi$ in Table 2(a).

Therefore, in Table 5(a), ^(a) both these are possible; which is indicated by the mixed state $\vee \oplus \xi = \exists$. In the same way,

in Table 2(a), ^(a) the output Φ for \underline{c}' occurs in the second row for the second column (\wedge) and third column (Φ), both of which are

possible solutions. The Boolean sum of these two is once again $\wedge : (\Phi \oplus \wedge) = (0\ 0\ 1) \oplus (0\ 1\ 1) = (0\ 1\ 1) = \wedge$, ^{for (ξ, Θ) in Table 5(a).} This completes the description of Table 5(a).

Table 5(d) is similar, and is in fact simpler to work out.

The first and the third rows are automatic, since there is only

one value of \underline{b}' that leads to each generator state for \underline{c}' and it is interesting that the corresponding rows 1 and 3 of Table 2(d) and Table 5(d) are identical. In fact, this identity extends also to row 2, but the explanation of this is not so obvious. Thus, taking $\underline{a}' = \underline{\zeta}$ and $\underline{c}' = \underline{V}$, the latter occurs only in the entry Δ for $(\underline{\zeta}, \underline{\zeta})$ in Table 2(d). Consequently, the first entry in row 2 of Table 5 is $\underline{\zeta}$; so also the third entry is $\underline{\zeta}$. On the other hand, in Table 2(d), $\underline{\zeta}$ is contained in all three entries in the middle row, and all of them are operative for the reverse relation, leading to $\underline{c}' = \underline{\zeta}$ from $\underline{a}' = \underline{\zeta}$ — hence the $\Delta = (\underline{V} \oplus \underline{\zeta} \oplus \underline{\zeta})$ in the middle of the second row of Table 5(d).

Thus, it will be seen that we ^{have} / only to pick out the corresponding generator states that are provided by each entry in Table 2 and rearrange them so as to get Table 5. This has been done for the other two types of matrix connectives also and Tables 5(a-d) are the result. We shall show below that the complete 8x8 table for the four types of connectives ^{also} can be deduced from the 3x3 table for the generator states. In particular, the 4x4 table for the four standard quantifier

states will come as a portion of the 8x8 table.

The method of deriving the entries in Table 6 from those in Table 5 can be very generally stated as follows. Suppose we wish to find $\underline{b}'x$ of Eq.(36) given the data for $\underline{a}'x$, $\underline{c}'x$ and $\underline{Z}(1, 1)$ on its l.h.s. We express the quantifier states of $\underline{a}'x$ and $\underline{b}'x$ as Boolean sums of the generator states as in (37) below:

$$\underline{a}'x = \bigoplus_i q(g_i) \quad , \quad \underline{c}'x = \bigoplus_j q(g_j), \quad g_i, g_j = \begin{matrix} \text{generator} \\ \text{states} \end{matrix} \quad (37)$$

In Table 5, we have data for $\underline{a}'x(\underline{Z}, \underline{c}'x) = \underline{b}'x$ for the nine combinations (g_i, g_j) — namely pairs of generator states.

In order to obtain the corresponding entry $\underline{b}'x$ for the mixed states $\underline{a}'x, \underline{c}'x$ given in (37), we have only to list the pairs (g_i, g_j) , $(i, j = 1, 2, 3)$ which are present in the combination $(\underline{a}', \underline{c}')$, and make a Boolean sum of these, to obtain the output $\underline{b}'x$ as in (38):

$$\bigoplus_i \bigoplus_j q(g_i) (\underline{Z}, q(g_j)) \quad \underline{b}'x \quad (38)$$

We shall give the full tables and show therefrom that a theory of QL-1 relations based only on the four standard quantifier states is inadequate to completely work out the 8x8 tables for all the QBA states.

Table 6. Full 8x8 tables for QL-1 unary matrix connectives

$$(a) \quad \underline{a}'x \ (A, c'x) = b'x$$

$\underline{a}' \backslash \underline{c}'$	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
\forall	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
\exists	\forall^1	\exists	Δ	\wedge	\exists	Δ	Δ	ϕ
\wedge	ϕ	\exists^2	Δ	Δ	\exists	Δ	\wedge	ϕ
Φ	ϕ	ϕ	Δ	\wedge	ϕ	\wedge	\wedge	ϕ
Σ	ϕ	\exists	Δ	\wedge	\exists	Δ	\wedge	ϕ
Δ	\forall	\exists	Δ	\wedge	\exists	Δ	Δ	ϕ
Θ	\forall	\exists	Δ	Δ	Σ	Δ	Δ	ϕ
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

$$\underline{a}'_0 = \forall^1, \Sigma^1$$

$$(b) \quad \underline{a}'x \ (Q, c'x) = b'x$$

$\underline{a}' \backslash \underline{c}'$	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
\forall	Δ	Δ	ϕ	ϕ	ϕ	Δ	Δ	ϕ
\exists	Δ	Δ	\wedge^1	ϕ	\wedge	Δ	Δ	ϕ
\wedge	\exists	Δ	\wedge	Φ^2	\wedge	Δ	\wedge	ϕ
Φ	\forall	\exists	\wedge	Φ	Σ	Δ	Θ	ϕ
Σ	\exists	Δ	\wedge	ϕ	\wedge	Δ	\exists	ϕ
Δ	Δ	Δ	\wedge	Φ	\wedge	Δ	Δ	ϕ
Θ	Δ	Δ	\wedge	Φ	Σ	Δ	Δ	ϕ
ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

$$\underline{a}'_0 = \Sigma^1, \Phi^1$$

(c) QL-1 unary relation for all eight states of QBA

As mentioned above, the binary reverse relation obtained by reversing the binary forward relation (35) is completely equivalent to the standard QL-1 forward relation (34), in the form of Eq.(36). In Tables 5(a-d), we have listed in tabular form the output of the unary relation corresponding to the inputs $q(\underline{ia}'x)$ and $q(\underline{iz}x)$ having one or the other of the generator states, and indicated in Eqs (37) and (38) the way in which these tables can be extended to obtain the corresponding 8x8 tables. These are shown in Table 6(a-d). We shall describe below some consistency checks that can be made on these tables, which will assure that they agree with standard logical expectations which can be worked out from apriori considerations. At the same time, it will be found that it is not possible to define the state "there exists, but not for all" as $\exists \otimes \wedge$ and carry this through for the unary relation as well — the meaning of this will be clear as we go along. On the other hand, both \exists and \wedge can be defined in terms of \vee , \sum , $\bar{\Phi}$ as suitable Boolean sums of the generator states, and this can be carried through without any difficulty. In other words, we shall show that

the algebra requires apriori definition of the three QBA generator states and their properties.

(i) Logical checking of Table 6(a) for \underline{A}

Taking the first row, it is obvious that, if one of the inputs in the QL-1 forward binary relation $\underline{a}'x \underline{A} \underline{b}'x = \underline{c}'x$ is $\underline{a}'x = \forall$, then the output $\underline{c}'x$ determines the nature of the other input $\underline{b}'x$ completely, so that the entries in the first row correspond completely to those in the top of the table.

The second row corresponding to $\underline{a}'x = \exists$ is rather interesting. The first two entries for (\exists, \forall) and (\exists, \exists) , namely \forall and \exists , for $\underline{b}'x$, show that the output of the unary relation $q(\underline{z}x)(\underline{a}x \underline{A} \underline{b}x)$ depends only on the quantifier state of $\underline{z} = \underline{c}'x$ and not on that of $\underline{a}'x$. To make the position clear, we shall consider the following two examples.

$$(\exists x)(\underline{s}x), (\forall x)(\underline{s}x \wedge \underline{i}x) \mapsto (\forall x)(\underline{i}x), \quad \begin{array}{l} \underline{s} = \text{student} \\ \underline{i} = \text{intelligent} \end{array} \quad (39a)$$

and

$$(\exists x)(\underline{s}x), (\exists x)(\underline{s}x \wedge \underline{i}x) \mapsto (\exists x)(\underline{s}x), \quad \begin{array}{l} \underline{s} = \text{student} \\ \underline{i} = \text{intelligent} \end{array} \quad (39b)$$

It will be seen that the input, namely "there exists students" does not at all determine the output of the relation containing

a conjunction, because the conjunction itself is a declaratory statement saying that $(\forall x)(\underline{ix})$ is true in (39a) and $(\exists x)(ix)$ is true in (39b). So also, in the first row, the ent corresponding to (\forall, \exists) takes a similar form

$$(\forall x)(\underline{sx}), (\exists x)(\underline{sx} \wedge \underline{ix}) \longleftrightarrow (\exists x)(\underline{ix}) \quad (39c)$$

As is normally noticed, in most treatments on AL-1 statements the quantifier of the input is suitably absorbed into the quantifier of the relation so that one would find that the output for (39c) is the same as given therein while for 39(a and b) they are $(\exists x)(\underline{ix})$ and $(\Delta x)(\underline{ix})$ respectively (See MR-34, MR-45 etc .) On the other hand, an examination of the strict meaning of the quantifier as defined in ALOG-50 indicates that the assertion mentioned above, of the conjunction in a quantified statement, is mandatory and indicate the conclusions given in Table 6(a) can all be verified by the resolution method of checking theorems. Therefore, this table can be taken to be the correct formulation of AL-1 statements according to the definition of a quantifier state given via SNS relations in Section 1 and Table 1 of ALOG-50.

The entries in the first column in rows 3 and 4 readily follow

from the above considerations, since the inputs $\underline{a}'x = \underline{\Lambda}$ and $\underline{c}' = \underline{V}$, so that the contradictory state ϕ is the consequence; similarly for $(\underline{\Phi}, \underline{\exists})$. On the other hand, the entry $\underline{\exists}$ corresponding to $(\underline{\Lambda}, \underline{\exists})$ is rather interesting. While it is derivable from Table 2(a) for the generator states, it is not obvious at all from Table 3(a) containing the standard quantifier states, unless those data are reinterpreted in terms of the generator states. However, the fact that

$$(\underline{\Lambda} x)(\underline{a}'x), (\underline{\exists} x)(\underline{a}x \underline{A} = \underline{b}x) \mapsto (\underline{\exists} x)(\underline{b}'x) \quad (40)$$

can be shown to be valid from first principles. We shall give an example first and then indicate the derivation via the symbology

Consider the following statements.

$$(\underline{\exists} x)(\underline{\neg} \underline{d}'x) \equiv \text{Some members are not doctorates} \quad (41a)$$

$$(\underline{\exists} x)(\underline{d}'x \wedge \underline{e}'x) \equiv \text{There are some members who are both doctorates and executives.} \quad (41b)$$

It will be seen that the former statement (41a) does not influence the conclusion that

$$(\underline{\exists} x)(\underline{e}'x) \equiv \text{Some members are executives} \quad (41c)$$

This is the maximum that can be deduced about executives from the two statements (41a,b). In essence, the conjunction leads straightaway to the conclusion, independent of any statement made about the input (in this case doctorates), unless the input is contradictory to the conjunction. An examination of columns 1 and 2 of Table 6(a) shows that, in fact, this is fully satisfied. All entries in the first column are either \forall or ϕ and all entries in the second column are either \exists or ϕ . In both cases, the impossible state ϕ arises if/there is contradiction between the input and the relation.

It is interesting that a detailed logical examination leading to these results are automatically taken into account and the quantifier state of the conclusion given straightaway in Table 6/ by deriving these, using the/ in Table 5(a) for the generator states/ This is the great merit of the BVM formalism, in that the Boolean algebra/ (in this case QBA) incorporates the essential logical ideas, and, thereafter, one need not check the progress of the argument, and the conclusion can be worked out purely in terms of algebraic manipulations.

(a)

Columns 3 and 4 of Table 6 are however different.

This is because the quantifier states \wedge and \oplus for $c'x$ are to be interpreted as leading to the $QL=1$ relations

$(\exists x)(\neg \underline{a}x \underline{Q} \neg \underline{b}x)$ and $(\forall x)(\neg \underline{a}x \underline{Q} \neg \underline{b}x)$ respectively,

following from the equation $\underline{A}^c(1, 1) \equiv \underline{Q}(2, 2)$. Since an

\underline{Q} -type relation is implicational, there can never be any

contradiction, and this is indicated by the absence of the

state ϕ in these two columns. It can also be verified that,

corresponding to the above formulae involving disjunctions,

the relation

$$\underline{a}'n_x \underline{Q} \underline{b}'n_x = c'x \quad (42)$$

is satisfied by all the entries in these columns. In other

words, without ever positively stating that the logical

connective \underline{Q} of disjunction is operative in these cases,

the BVM formulae take care of this fact, and the logical

property is incorporated in the entries, by expressing the

quantifiers in QBA, and reversing Table 2 to obtain Table 5,

and hence obtaining the data in Table 6.

It will be seen that the first four entries in rows

5, 6 and 7 for the inputs $\underline{\zeta}$, $\underline{\Delta}$, $\underline{\Theta}$ respectively, can be

implication via disjunctions and therefore no fresh comments are needed for Table 6(c).

Table 6(d), on the other hand, is invariant under the transformation mentioned above which carries over 6(a) to 6(b), and therefore another table cannot be derived by interchanging the truth values T and F from Table 6(d). Its properties are fairly clear, except for the comment mentioned earlier regarding the non-permissibility of Eq.(43a) in general, which will be commented upon below. Otherwise, (43b,c,d) hold equally well for Table 6(d) also.

(iii) Comments on the right hand half of Tables 6(a-d)
and occurrence of two inconsistencies in each.

Exactly similar to (43a-d), one can obtain (44a-d) below by taking Boolean sums and products of the four standard quantifier states, at the top of the table, to obtain the remaining four in the right half.

$$\underline{p}'(q(\underline{1}), \underline{\exists}) \quad \underline{p}'(q(\underline{1}), \underline{\exists}) \otimes \underline{p}'(q(\underline{1}), \underline{\wedge}) \quad (44a)$$

$$\underline{p}'(q(\underline{1}), \underline{\Delta}) \quad \underline{p}'(q(\underline{1}), \underline{\vee}) \oplus \underline{p}'(q(\underline{1}), \underline{\wedge}) \quad (44b)$$

$$\underline{p}'(q(\underline{1}), \underline{\exists}) \oplus \underline{p}'(q(\underline{1}), \underline{\exists}) \quad (44c)$$

$$\underline{p}'(q(\underline{1}), \underline{\ominus}) \quad \underline{p}'(q(\underline{1}), \underline{\vee}) \oplus \underline{p}'(q(\underline{1}), \underline{\Phi}) \quad (44d)$$

It can be verified that all these equations are readily satisfied for the upper half of each of the tables 6(a-d).

We find an inconsistency only in the bottom right hand quarter in each of these tables. We shall first mention these, and then give a possible explanation of why they occur. Thus, taking Table 6(a), the entry for $\underline{p}'(\underline{\xi}, \Theta)$ can be obtained in two ways, by combining two of the equations in (43a-d), and (44a-d), as follows:

$$\underline{p}'(\underline{\xi}, \Theta) = \underline{p}'(\underline{\xi}, \underline{V}) \oplus \underline{p}'(\underline{\xi}, \underline{\Phi}) = \emptyset \oplus \Lambda = \Lambda \quad (45a)$$

$$\underline{p}'(\underline{\xi}, \Theta) = \underline{p}'(\underline{\exists}, \Theta) \otimes \underline{p}'(\Lambda, \Theta) = \Delta \otimes \Delta = \Delta \quad (45b)$$

It is obvious that (45a,b) do not agree. Only the former agrees with the correct value Λ , obtained from first principles, using the generator states \underline{V} , $\underline{\xi}$ and $\underline{\Phi}$. Since the latter employs the Boolean product, some difficulty associated with the failure of either the commutational, or the associative, properties of the Boolean sum and product has intervened in producing the error.

In fact, a similar difficulty arises also for $\underline{p}'(\Theta, \underline{\xi})$ of Table 6(a), where the formula with the Boolean sum is correct, while the Boolean product leads to a wrong value. Also, the

implication via disjunctions and therefore no fresh comments are needed for Table 6(c).

Table 6(d), on the other hand, is invariant under the transformation mentioned above which carries over 6(a) to 6(b), and therefore another table cannot be derived by interchanging the truth values T and F from Table 6(d). Its properties are fairly clear, except for the comment mentioned earlier regarding the non-permissibility of Eq.(43a) in general, which will be commented upon below. Otherwise, (43b,c,d) hold equally well for Table 6(d) also.

(iii) Comments on the right hand half of Tables 6(a-d and occurrence of two inconsistencies in each.

Exactly similar to (43a-d), one can obtain (44a-d) below by taking Boolean sums and products of the four standard quantifier states, at the top of the table, to obtain the remaining four in the right half.

$$\underline{p}'(q(\underline{i}), \underline{\exists}) = \underline{p}'(q(\underline{i}), \underline{\exists}) \otimes \underline{p}'(q(\underline{i}), \underline{\wedge}) \quad (44a)$$

$$\underline{p}'(q(\underline{i}), \underline{\Delta}) = \underline{p}'(q(\underline{i}), \underline{\vee}) \oplus \underline{p}'(q(\underline{i}), \underline{\wedge}) \quad (44b)$$

$$= \underline{p}'(q(\underline{i}), \underline{\exists}) \oplus \underline{p}'(q(\underline{i}), \underline{\exists}) \quad (44c)$$

$$\underline{p}'(q(\underline{i}), \underline{\Theta}) = \underline{p}'(q(\underline{i}), \underline{\vee}) \oplus \underline{p}'(q(\underline{i}), \underline{\Phi}) \quad (44d)$$

same difficulty is found in Tables 6(b,c,d) also. Of these, the problem in Tables 6(b) and 6(c) is not novel, since we have shown above that the three tables 6(a,b,c) are inter-related. However, the one in 6(d) is worth noting. As a matter of fact, the same type of trouble is noticed even in Table 2(d), for the output \underline{c}' of the QL-1A binary forward relation $\underline{a}'x \underline{E} \underline{b}'x = \underline{c}'x$. This had been noticed earlier, but was not commented upon as it was a single exception in Table 2(d), while this did not happen in Tables 2(a,b and c). Now that it arises in all the tables in 6(a-d), it must be considered to be a serious difficulty. It will be noticed that it arises out of the fact that the resultant output is for a relation, in which the generator states of the quantifier terms are combined with one another both by the Boolean sum and the Boolean product, and the latter is applied after the former is applied. On the other hand, if $\underline{\Sigma}$ is taken as a generator state, and the Boolean product is not employed between \exists and \wedge for working out the properties of $\underline{\Sigma}$, then no inconsistency arises.

The above considerations will indicate that the theory of logical relations in QL-1 should be formulated in terms of the generators of the BA-3 algebra of QBA, rather than

based upon the two standard quantifier states \forall and \exists , and representing the other two quantifier states \wedge and ϕ in terms of them, and then obtaining all possible quantifier states in terms of these four. The difficulty mentioned in the previous subsection is quite serious, and it cannot be assured that a general statement in QL-1 is correctly worked out if it is implemented only in terms of the four standard quantifier states. However, the use of the Boolean sum to give the properties of a mixed state from those of the generator state is such a fundamental and basic principle in BVMF, that there can be no difficulty, if this is adhered to all throughout.

(d) Implementation of QL-1 unary relations for $\underline{Z}(k, \ell)$

So far we have only considered the utilization of the QL-1 connectives $\underline{A}(1, 1)$, $\underline{Q}(1, 1)$, $\underline{I}(1, 1)$ and $\underline{E}(1, 1)$ for working out the output of a QL-1 unary relation. We shall now extend this for a general $\underline{Z}(k, \ell)$ as follows:

Consider the relation

$$\underline{a}'x, \quad q(\underline{i}Zx) \quad (\underline{a}x \quad \underline{Z}(k, \ell) = \underline{b} \ x) \quad \mapsto \quad \underline{b}'x \quad (46a)$$

Denoting the quantifier states of the input, the connective and the output by $q(\underline{i}a'x)$, $q(\underline{i}c'x)$ and $q(\underline{j}b'x)$ respectively,

this takes the form

$$q(\underline{ia}'x) (Z(k, \underline{\ell}), q(\underline{ic}'x)) = q(\underline{jb}'x) \quad (46b)$$

The implementation of (46b) is done in three steps as follows:

Calculate

$$q(\underline{ia}''x) = q(\underline{ia}'x) \text{ or } q(\underline{i}^n a''x), \text{ according as } k = 1, 2 \quad (47a)$$

$$q(\underline{ia}''x) (Z(1, 1), q(\underline{ic}'x)) = q(\underline{jb}''x), \quad Z = \underline{A}, \underline{O}, \underline{I}, \underline{E} \quad (47b)$$

$$q(\underline{jb}'x) = q(\underline{jb}''x) \text{ or } q(\underline{j}^n b''x), \text{ according as } \underline{\ell} = 1, 2 \quad (47c)$$

In this way, the output $\underline{b}'x$ for a general Q^{L-1} relation (46a) can be worked out for a given input $a'x$.

(i) Consistency check of the input against the relation and its consequent modification

We have not considered so far the interaction of the relation on the input in a general manner, except pointing out that, under certain conditions, the two may be contradictory, when the output $\underline{b}'x$ becomes the impossible state (\varnothing) . When this happens, a consistency check must be made between the input and the quantifier state of the relation, involving both the actual quantifier state $q(\underline{iz}x)$ and the SNS state $s(k)$.

Following what was discussed in the earlier sections, this takes the form of the vidya check (48b) below. Define

$$q(\underline{i}Z'x) = q(\underline{i}Zx) \text{ or } q(\underline{i}^nZx), \text{ according as } k = 1 \text{ or } 2 \quad (48a)$$

Calculate the vidya product (48b) for the left half of the 4x4 sub-table in Table 6(a) for $\underline{Z} = \underline{A}$ and r.h.s of Tables 6(b) and 6(c) for $\underline{Z} = \underline{Q}$ or \underline{I} , to give the revised input $q(\underline{ia}'_0x)$:

$$q(\underline{ia}'x) \vee q(\underline{i}Z'x) = q(\underline{ia}'_0x) \quad (48b)$$

If $q(\underline{ia}'_0x) = \phi$, it confirms the contradiction as arising from that between the input and the relation. If it is not equal to ϕ , but different from $q(\underline{ia}'x)$, the revised state $q(\underline{ia}'_0x)$ is the net resultant of the information contained in both the input and the relation. Otherwise, there is no need to revise the input.

This possibility has been worked out in detail for the left hand top 4x4 sub-tables of Tables 6(a-d) and are given in the footnotes to these tables. For all other entries in the sub-tables, there is no need for a revision of the input.

6. Multiply quantified relations in QL-1

In the previous sections 3 and 5, we have given the formulae for the implementation of a QL-1 binary forward relation, and of a QL-1 binary reverse relation, which is equivalent to a unary relation. These were for a single

elementary relation in one variable x . In this section, we shall extend these formulae to multiply quantified QL-1 relations, both of the binary forward and binary reverse (unary) relations . We shall not describe the procedure to obtain the essential formulae, which follow from the corresponding formulae in the two sections mentioned above.

(a) Unary relation

The problem can be stated as follows:

Given the multiply quantified relation (49a) and the input in (49b), we wish to obtain the output in (49c):

Relation

$$q(\underline{i}Zx) q(\underline{i}Zy) \dots q(\underline{i}Zv) (\underline{a}xy \dots \vee \underline{z}(k, \ell) = \underline{b}xy \dots v) \quad (49a)$$

Input

$$q(\underline{i}a'x) q(\underline{i}a'y) \dots q(\underline{i}a'v) (\underline{a}'xy \dots v) \quad (49b)$$

Output

$$q(\underline{j}b'x) q(\underline{j}b'y) \dots q(\underline{j}b'v) (\underline{b}'xy \dots v) \quad (49c)$$

The procedure for this is to obtain each one of the quantifier states in the multiply quantified output, from the corresponding quantifiers in the input, and the relation, and then combining them to obtain the output. Thus for $u = x, y, \dots, v$, the

following formulae (50a,b,c) give the $q(\underline{j}b'u)$, from $q(\underline{i}a'u)$, $q(\underline{i}Zu)$ and the relational matrix $\underline{Z}(k, \ell)$, employing the QL=1 connective $\underline{Z}(1, 1)$. These three equations follow completely from (47a,b,c).

$$q(\underline{i}a''u) = q(\underline{i}a'u) \text{ or } q(\underline{i}^n a'u), \text{ according as } k = 1 \text{ or } 2 \quad (50a)$$

$$q(\underline{i}a''u) (\underline{Z}(1, 1), q(\underline{i}Zu)) = q(\underline{j}b''u) \quad (50b)$$

$$q(\underline{j}b'u) = q(\underline{j}b''u) \text{ or } q(\underline{j}^n b''u), \text{ according as } \ell = 1 \text{ or } 2 \quad (50c)$$

It will be noted that we take both \underline{a}' and \underline{b}' , forming the predicates of the input and the output, to have the SNS state T and all the information regarding the ^{two} ^{terms} /quantified/ is contained in (49b) and (49c).

It is implicitly assumed that all the variables x, y, \dots, v are the same for the input and the output and that all are present in both of them. We only generalize from a single variable to multiple variables. The extension to multiply ^{terms is} /quantified/ straightforward for the unary relation; but it requires some extra manipulations for the binary relation, as given below.

(b) Binary forward relation

In this case, the relation has a form which is a generalization of Eq.(21), and can be stated as in (51a - e):

Inputs

$$\underline{a}' = q(\underline{ia}'x) q(\underline{ia}'y) \dots q(\underline{ia}'v)(\underline{a}'xy \dots v) \quad (51a)$$

$$\underline{b}' = q(\underline{jb}'x) q(\underline{jb}'y) \dots q(\underline{jb}'v)(\underline{b}'xy \dots v) \quad (51b)$$

Relation

$$q(\underline{iz})(\underline{a} \underline{z}(k, \ell) \underline{b} = \underline{c}); q(\underline{iz}) = q(\underline{iz}x) q(\underline{iz}y) \dots q(\underline{iz}v) \quad (51c)$$

Output

$$\underline{c}' = q(\underline{ic}'x) q(\underline{ic}'y) \dots q(\underline{ic}'v)(\underline{c}'xy \dots v) \quad (51d)$$

$$\text{and} \quad \underline{t}(\underline{z} \mid \underline{a}', \underline{b}') \quad (51e)$$

Just as for the unary relation, we first calculate each $\underline{t}(u)$ for $u = x, y, \dots, v$ and then obtain from them the resultant truth value/ (51e) full of the relation for the given inputs as given by (51e). This is done as follows. For each u calculate (52a,b,c) :

$$q(\underline{ia}''u) = q(\underline{ia}'u) \text{ or } q(\underline{i}^n a''u), \text{ according as } k = 1, 2 \quad (52a)$$

$$q(\underline{jb}''u) = q(\underline{jb}'u) \text{ or } q(\underline{j}^n b''u), \text{ according as } \ell = 1, 2 \quad (52b)$$

$$q(\underline{ia}''u) \underline{z}(1, 1) q(\underline{jb}''u) = q(\underline{ic}'u) \quad (52c)$$

Thereafter the values of $q(\tilde{ic}'u)$ can be taken over to give the output \tilde{c}' in (51d). In order to calculate the truth value in (51e), we make the calculations in (53a,b):

$$\underline{t}(q(\tilde{iz}u) \mid q(\tilde{ic}'u)) = \underline{t}(u) , u = x, y, \dots, v \quad (53a)$$

$$\underline{t}(x) \underline{A} \underline{t}(y) \dots \underline{A} \underline{t}(v) = \underline{t}(\underline{z} \mid \underline{a}', \underline{b}') \quad (53b)$$

In this way, the formulae for a single variable QL-1 binary relation, developed in Section 3, can be extended for a multiply quantified relation also. The possibility of negated multiple quantifier, either for the relation, or for input terms, is not discussed in particular. So also, the checking for contradiction in the case of unary relations is not described. Both these follow the principles adopted for singly quantified QL-1 relations, and are extendable, following the lines adopted for a general multiply quantified QL-2 relation .

In the same way, the occurrence of absent variables and non-quantified variables is not discussed here. One aspect of this is considered in the following section dealing with interconversion of QL-1 and QL-2 types of relations which is highly relevant for first order predicate logic — for instance

in the conversion of a genral multiply quantified statement into its prenex normal form. We shall indicate in our approach how such sentences can be put in the QL-2 form, which is generally implementable, using QL-2' formalism, and QBA states for quantifiers.

(c) Interconversion of QL-1 and QL-2 forms of relations in predicate logic

(i) Case with one term without variable

First we shall prove the following formulae.

$$q(\underline{i}x)(\underline{a}x \underline{\leq} \underline{b}) \equiv q(\underline{i}x)(\underline{a}x) \underline{\leq} \underline{b}, \underline{\leq} = \underline{A}(1, 1), \underline{Q}(1, 1), \underline{E}(1, 1) \quad (54a)$$

$$q(\underline{i}x)(\underline{a} \underline{\leq} \underline{b}x) \equiv \underline{a} \underline{\leq} q(\underline{i}x)(\underline{b}x), \underline{\leq} = \underline{A}(1, 1), \underline{Q}(1, 1), \underline{E}(1, 1) \quad (54b)$$

These are well-known formulae in standard predicate logic, and are proved for \underline{a} and \underline{b} having BA-1 truth values. We shall extend the proof to BA-2 truth values and to QBA states for $q(\underline{i}x)$. In this case, we need prove them only for the three generator states $\underline{i} = \underline{1}, \underline{3}, \underline{5}$ for $q(\underline{i}x)$ in (54a) and (54b) respectively. These generator states are defined in Table 1.

We therefore assume the equivalences

$$\begin{aligned} (\forall x)(\underline{a}x) &\equiv \bigoplus_x \underline{a}x = T \text{ for } q(\underline{1}), (\Phi x)(\underline{a}x) \equiv \bigoplus_x (\underline{a}x) = F \text{ for } q(\underline{5}), \\ (\Sigma x)(\underline{a}x) &\equiv \bigoplus_x \underline{a}x = D, \text{ but not } T \text{ or } F \text{ for } q(\underline{3}) \end{aligned} \quad (55a,b,c)$$

These are BA-3 definitions of the QBA generator states \forall, Σ, Φ standing for TT, SS, FF in column 6, which are given, however, in BA-2 language, as in column 5 of Table 1.

We shall consider each of the three generator states separately for Eq.(54a).

$$q(1) = (\forall x)$$

$$A(1, 1)$$

In this case, the l.h.s and r.h.s of (54a) become

$$(\underline{ax} \underline{A} \underline{b}) = T, \text{ and } \bigoplus_x (\underline{ax}) \underline{A} \underline{b} = T \quad (56a,b)$$

respectively. Considering the former, every $(\underline{ax} \underline{A} \underline{b}) = T$ and therefore every \underline{ax} and \underline{b} are both equal to T. Consequently, in the latter, $\bigoplus_x (\underline{ax}) = T$ and $\underline{b} = T$, so that it is also valid. Thus (56b) follows from (56a)

Vice-versa, (56a) also follows from (56b), because (54) requires that $\underline{b} = T$ and every $\underline{ax} = T$, so that every $\underline{ax} \underline{A} \underline{b} = T$ and (56a) follows. Hence we the equivalence

$$(\forall x)(\underline{ax} \underline{A} \underline{b}) \equiv (\forall x)(\underline{ax}) \underline{A} \underline{b}, \text{ for } q(1) \quad (57a,b)$$

$$Q(1, 1)$$

In this case, we have, similar to (56a,b), the equations

$$x (\underline{ax} \underline{Q} \underline{b}) = T \text{ and } \bigoplus_x (\underline{ax}) \underline{Q} \underline{b} = T \quad (58a,b)$$

However, \underline{b} is not necessarily equal to T, and therefore, we shall consider the two possibilities of $b = T$ and $b = F$.

b = F : Eq.(58a) demands that every $(\underline{ax} \underline{\underline{0}} \underline{b}) = T$, and, since b = F, every $\underline{ax} = T$, so that $\bigoplus_x \underline{ax} = T$ in (58b). Consequently, (58b) is equal to T, and (58b) follows from (58a). Vice-versa, assuming (58b) to be true and b = F, it follows that $\bigoplus_x \underline{ax} = T$ so that every $\underline{ax} = T$. Consequently every $(\underline{ax} \underline{\underline{0}} \underline{b}) = T$ in (58a), and the multiple sum in (58a) is also T, and (58a) follows from (58b), making the two equivalent

b = T : In this case, starting from (58a), since every $(\underline{ax} \underline{\underline{0}} \underline{b}) = T$ and b = T, \underline{ax} can be T, F or D so that $\bigoplus_x \underline{ax} = T, F \text{ or } D$. However, $\bigoplus_x (\underline{ax}) \underline{\underline{0}} \underline{b} = (T, F, D) \underline{\underline{0}} T = T$, so that (58b) follows from (58a). Vice-versa, starting from (58b), since b = T, $\bigoplus_x \underline{ax} = T, F \text{ or } D$, so that every \underline{ax} can be T, F or D. However, for every x, $(\underline{ax} \underline{\underline{0}} \underline{b}) = T$ so that $\bigcup_x (\underline{ax} \underline{\underline{0}} \underline{b}) = T$ making (58a) derivable from (58b).

Thus, for both b = F and b = T, (58a) and (58b) are shown to be equivalent in the form

$$(\forall x)(\underline{ax} \underline{\underline{0}} \underline{b}) \equiv (\forall x)(\underline{ax}) \underline{\underline{0}} \underline{b} \quad \text{for } q(1) \quad (59a,b)$$

E(1, 1)

The argument is somewhat different in this case. In (54a),

the l.h.s and r.h.s become

$$\bigoplus_x (\underline{ax} \underline{\underline{E}} \underline{b}) = T \quad \text{and} \quad \bigoplus_x (\underline{ax}) \underline{\underline{E}} \underline{b} = T \quad (54a,b)$$

Hereagain, we consider the two cases $\underline{b} = T$ and $\underline{b} = F$.

$\underline{b} = T$: For this, (60a) demands that every $\underline{ax} = T$, so that

$$\bigoplus_x \underline{ax} = T, \text{ which, along with } \underline{b} = T, \text{ leads to (60b) being true.}$$

Vice-versa, if (60b) is true. $\bigoplus_x \underline{ax} = T$, so that every $\underline{ax} = T$, and every $(\underline{ax} \underline{\underline{E}} \underline{b}) = T$, leading to (60a) being true.

$\underline{b} = F$: For this, (60a) demands that every $\underline{ax} = F$, so that

$$\bigoplus_x \underline{ax} = F, \text{ which, with } \underline{b} = F, \text{ leads to (60b) being F. Vice-versa,}$$

if (60b) is T, then $\bigoplus_x \underline{ax} = F$, which requires that every $\underline{ax} = F$, so that every $(\underline{ax} \underline{\underline{E}} \underline{b}) = T$, and (60a) is satisfied. In this

case also, we obtain the equivalence of the r.h.s and l.h.s of (54a) in the form

$$(\forall x)(\underline{ax} \underline{\underline{E}} \underline{b}) = (\forall x)(\underline{ax}) \underline{\underline{E}} \underline{b} \quad (54c,d)$$

$$\underline{q(5)} = (\bigoplus x)^*$$

The above argument in connection with $\underline{q(1)} = (\forall x)$ can be taken over with suitable modifications for this case. It will be found that for $\underline{A(1, 1)}$, \underline{b} can be T or F, and in either case

*The proof for this case for $\underline{A(1, 1)}$ and $\underline{Q(1, 1)}$ is probably not valid and requires modification. Consequently, the general theory in this section (c) starting from page 59 needs changes. It is left as it is since the arguments in the revision have arisen out of the ideas written here.

the equivalence

$$(\Phi x)(\underline{ax} \underline{A} \underline{b}) \equiv (\Phi x)(\underline{ax}) \underline{A} \underline{b} \quad (62a,b)$$

is satisfied. However, for $Q(1, 1)$, it will be seen that, since $(\Phi x)(\underline{ax})$ requires that $\bigvee_x \underline{ax} = F$, so that every \underline{ax} is F, \underline{b} can only be F, whether the l.h.s or the r.h.s is taken to be valid, and the other side follows unequivocally.

E(1, 1)

to that for $q(1)$,

The argument is very similar/with the difference that

$\underline{b} = F$ and $\underline{b} = T$ exchange their roles, and the former proof for $q(1)$ is valid for the latter case of $q(2)$, and vice-versa.

In either case, it follows that

$$(\Phi x)(\underline{ax} \underline{E} \underline{b}) \equiv (\Phi x)(\underline{ax}) \underline{E} \underline{b} \quad (63a,b)$$

$$q(3) = (\Sigma, x)$$

A(1, 1)

The l.h.s and r.h.s of (54a) become as follows:

$$\bigoplus_x (\underline{ax} \underline{A} \underline{b}) = D, \text{ but not } T \text{ or } F ;$$

$$(\underline{ax}) \underline{A} \underline{b} = D, \text{ but not } T \text{ or } F \quad (64a,b)$$

Considering (64a) first, since the multiple sum cannot be F, there must be at least one x for which $(\underline{ax} \underline{A} \underline{b}) = T$, so that there is at least one x for which $\underline{ax} = T$, and $\underline{b} = T$ absolutely. Consequently, in (64b), $\bigoplus \underline{ax} = D$, but not T or F and $\underline{b} = T$, so that the conjunction of the two is but not T or F as required. Thus (64b) follows from (64a).

Vice-versa, if (64b) is true, \underline{b} may be T or F, but since the conjunction is only D, but not T or F, \underline{b} cannot be F, so that as in the above argument $\underline{b} = T$. Then, at least one $\underline{ax} = F$ and one $\underline{ax} = T$, which requires, in (64a), that at least one $(\underline{ax} \underline{A} \underline{b}) = T$ and one $(\underline{ax} \underline{A} \underline{b}) = F$, which is the necessary condition for (64a) to be true.

Consequently we have proved the equivalence of (64a) in the form

$$(\sum x)(\underline{ax} \underline{A} \underline{b}) \equiv (\sum x)(\underline{ax}) \underline{A} \underline{b} \quad (64a, b)$$

$\underline{Q}(1, 1)$

The above argument for $\underline{A}(1, 1)$ can be modified for this case, and it will be found that \underline{b} can only be F, and all the rest of the argument is valid, so that we get the equivalence

$$(\sum x)(\underline{ax} \underline{Q} \underline{b}) = (\sum x)(\underline{ax}) \underline{Q} \underline{b}.$$

E(1, 1)

We have for l.h.s and r.h.s of (54a),

$$\bigoplus_x (\underline{ax} \underline{E} \underline{b}) = D \text{ but not } T \text{ or } F ;$$

$$x \quad (\underline{ax}) \underline{E} \underline{b} = D, \text{ but not } T \text{ or } F \quad (66a, b)$$

(66a) requires that at least one $(\underline{ax} \underline{E} \underline{b}) = T$ and at least one $(\underline{ax} \underline{E} \underline{b}) = F$. Here, we must consider both $\underline{b} = T$ and $\underline{b} = F$.

$\underline{b} = T$: Then at least one $\underline{ax} = T$ and at least one $\underline{ax} = F$, so that $\bigoplus_x \underline{ax} = D$, but not T or F and (66b) is satisfied.

$\underline{b} = F$: Once again, at least one $\underline{ax} = F$ and at least one $\underline{ax} = T$, which is equivalent to what was derived above, so that $\underline{ax} = D$ but not T or F , and the r.h.s of (66b) is satisfied.

Taking (66b) we again consider the two cases $\underline{b} = T$ and F .

$\underline{b} = T$ This require that $\bigoplus_x \underline{ax} = D$ but not T or F , so that at least one $\underline{ax} = T$ and at least one $\underline{ax} = F$, which brings the same condition for $(\underline{ax} \underline{A} \underline{b})$, so that $\bigoplus_x (\underline{ax} \underline{A} \underline{b}) = D$, but not T or F , as in (66a).

$\underline{b} = F$: As in the previous case, this also reduces to the same proof, and (66a) follows from (66b).

Consequently, for $\underline{E}(1, 1)$ also $(\sum x)(ax \underline{E} b) = (\sum x)(ax) \underline{E} b$

Thus, for all three connectives $\underline{A}(1, 1)$, $\underline{Q}(1, 1)$, $\underline{E}(1, 1)$ we have shown the equivalence of l.h.s and r.h.s of (54a) for the three generator states $q(\underline{1})$, $q(\underline{3})$ and $q(\underline{5})$, from which it follows from the ^{algebra of} BA-3 / QBA that the equivalence also holds for all the 8 QBA states.

from that of (54a),

The equivalence in (54b) follows from the commutative property of all the three connectives, which is obvious from the symmetry of their 2x2 matrices:

$$\underline{A}(1, 1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad \underline{Q}(1, 1) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \quad \underline{E}(1, 1) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (67a)$$

It should be noticed that ^{this} does not hold for $\underline{I}(1, 1)$ whose matrix is not symmetric, namely

$$\underline{I}(1, 1) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (67b)$$

In fact, (54a) does not hold for $\underline{I}(1, 1)$ although (54b) does (see below).

The extension of (54a,b) to $Z = \underline{I}$ can be done from the definition of $\underline{I}(1, 1)$ as

$$ax \underline{I} b = \neg ax \underline{Q}(1, 1) b \quad (68)$$

Then, using (54a) we have

$$\begin{aligned} q(\underline{j}ax)(\underline{ax} \underline{\text{I}} \underline{b}) &= q(\underline{j}ax)(\neg \underline{ax} \underline{\text{O}} \underline{b}) \\ &= q(\underline{j}ax)(\neg \underline{ax}) \underline{\text{O}} \underline{b} = \neg q(i^{\underline{L}}ax) \underline{\text{O}} \underline{b} \equiv q(i^{\underline{L}}ax)(\underline{ax}) \underline{\text{I}} \underline{b} \end{aligned} \quad (69)$$

In standard symbols, it leads to the two well-known equations

$$(\forall x)(\underline{ax} \Rightarrow \underline{b}) \equiv (\exists x)(\underline{ax}) \Rightarrow \underline{b} ; \quad (70a)$$

$$(\exists x)(\underline{ax} \Rightarrow \underline{b}) \equiv (\forall x)(\underline{ax}) \quad \underline{b} \quad (70b)$$

In a similar manner, (54b) leads to an equation similar to (70), but of the form

$$q(\underline{j}bx)(\underline{a} \underline{\text{I}} \underline{bx}) \equiv \underline{a} \underline{\text{I}} q(\underline{j}bx)(\underline{bx}) \quad (71)$$

with no $i^{\underline{L}}$ as in (69). Therefore, in standard notation, the formulae obtainable from (71) are

$$(\forall x)(\underline{a} \quad \underline{bx}) \equiv \underline{a} \Rightarrow (\forall x)(\underline{bx}) \quad (72a)$$

$$(\exists x)(\underline{a} \quad \underline{bx}) \equiv \underline{a} \Rightarrow (\exists x)(\underline{bx}) \quad (72b)$$

In the above derivation of (54a) we have assumed that \underline{b} is an SMS term not associated with a variable. However, the exact condition for this equation to be valid is slightly more general and can be extended even to a quantified term $q(\underline{j}by)(\underline{by})$. This is considered in the next subsection.

(ii) Case with both terms having two different variables.

The extension of formulae (54a) and (54b) to this general case is straightforward, and we shall first give the formulae before discussing their properties.

$$q(\underline{i}ax) q(\underline{j}by) (\underline{ax} \underline{\underline{Z}} \underline{by}) \equiv q(\underline{i}ax)(\underline{ax}) \underline{\underline{Z}} q(\underline{j}by)(\underline{by}),$$

$$\underline{\underline{Z}} = \underline{\underline{A}}(1, 1), \underline{\underline{O}}(1, 1), \underline{\underline{E}}(1, 1) \quad (73a)$$

and equivalent to

$$q(\underline{i}^{\ell}ax)(\underline{ax}) \underline{\underline{I}} q(\underline{j}by)(\underline{by}), \text{ for } \underline{\underline{Z}} = \underline{\underline{I}}(1, 1) \quad (73b)$$

Put in this form, the proof of (73a) looks very simple, in that each quantifier having the variable x , or y , as the case may be, binds the corresponding term \underline{ax} , or \underline{by} , inside the bracket and this condition is made possible by the general definition of the quantifier state as a multiple sum in (55a,b,c). The reason why the operation does not hold for $\underline{\underline{I}}(1, 1)$ is what is evident in the derivation of Eq.(69). In this, $\underline{ax} \underline{\underline{I}} \underline{b} = \neg \underline{ax} \underline{\underline{O}} \underline{b}$ for the SNS term \underline{ax} , and $q(\underline{i}ax)(\underline{ax}) \underline{\underline{I}} \underline{b} = \neg q(\underline{i}^{\ell}ax)(\underline{ax}) \underline{\underline{O}} \underline{b}$ for a QL term \underline{ax} with \underline{i} different from \underline{i}^{ℓ} . In fact, this difference

arises because the negation of a quantifier is different from the negation of a predicate and $\text{in } q(\underline{ix})(\neg ax) \equiv \neg q(\underline{i}^{\ell}ax)$, \underline{i}^{ℓ} is different from \underline{i} . Consequently, $q(\underline{ia}x)$ in the l.h.s of (73a) becomes $q(\underline{i}^{\ell}x)$ in QL-2 form (73b).

(iii) Interconversion of QL-1 and QL-2 forms of statements in predicate logic

As in the case of the SNS term \underline{b} not associated with a variable, it is implicitly assumed that the quantified term $q(\underline{jby})(\underline{by})$ does not contain the variable x in an unbound form. This is an essential condition for the quantifier $q(\underline{ia}x)$ to be capable of being taken inside and outside the bracket in (73a). Therefore, each quantifier $q(\underline{ia}x)$, $q(\underline{iby})$ respectively binds the corresponding term in the predicate within the bracket. In this method of formulating multiply quantified QL-1 statements, we assume that the exact sequence of the quantifiers is not important, and that the quantifier corresponding to each term within the bracket is automatically selected by the equality of the variable associated with the SNS term and its corresponding quantifier. On the other hand, the logical relations between the terms in the predicate should be satisfied by the sequence and the occurrence of

brackets in the description of the predicate. With these restrictive conditions, the quantifiers can be freely taken inside and outside the bracket containing the predicate, for the connectives $\underline{A}(1, 1)$, $\underline{Q}(1, 1)$ and $\underline{E}(1, 1)$. Therefore, if implications occur, the best method of converting a QL-1 type of relation into a QL-2 relation, in such cases, is to convert the implication into a disjunction, convert it to the other type and change it thereafter into the implication form, as shown in (69), and employed in (73b).

We shall briefly indicate the validity of the extension of the ideas in the previous subsection (i) to the equations contained in (73a) where \underline{by} is a quantified term. To show its correspondence to (54a and b) we shall write them as below;

$$q(\underline{iax})(\underline{ax} \underline{\underline{z}} \underline{by}) \equiv q(\underline{iax})(\underline{ax}) \underline{\underline{z}} \underline{by} \equiv \underline{ax} \underline{\underline{z}} \underline{by} \quad (74a)$$

$$q(\underline{jby})(\underline{ax} \underline{\underline{z}} \underline{by}) \equiv \underline{ax} \underline{\underline{z}} q(\underline{jby})(\underline{by}) \equiv \underline{ax} \underline{\underline{z}} \underline{by} \quad (74b)$$

In each case, the first statement of the QL-1 form has been converted to the third statement which is of the QL-2 form. This interchange between QL-1 and QL-2 forms will always be possible so long as the terms \underline{ax} and \underline{by} do not have any variables in

common. In order to distinguish between the two types of QL-1 statements, namely QL-1A which cannot be converted to QL-2 form, and QL-1B which can be converted as in (74), we give the standard forms of the two cases below.

QL-1A

Singly quantified:

$$q(\underline{i}Zx)(\underline{a}x \underline{Z} \underline{b}x) ; q(\underline{i}Zx) = \text{quantifier of the relation} \quad (75a)$$

Multiply quantified:

$$q(\underline{i}Zx) \dots q(\underline{i}Zw) (\underline{a}x \dots w \underline{Z}(k, \ell) \underline{b}x \dots w) \quad (75b)$$

QL-1B

Singly quantified:

$$q(\underline{i}ax)(\underline{a}x \underline{Z} q(\underline{i}by)(\underline{b}y)) \equiv q(\underline{i}ax)(\underline{a}x) \underline{Z} q(\underline{i}by)(\underline{b}y), \\ \underline{Z} = \underline{A}, \underline{O}, \underline{E} \quad (76a)$$

Multiply quantified:

$$q(\underline{i}ax) \dots q(\underline{i}aw) q(\underline{i}by) \dots q(\underline{i}bv) (\underline{a}x \dots w \underline{Z}(k, \ell)) (\underline{b}y \dots v) \\ q(\underline{i}ax) \dots q(\underline{i}aw) (\underline{a}x \dots x \underline{Z}(k, \ell)) q(\underline{j}by) \dots q(\underline{j}bv) \quad (76b)$$

We shall show that any general multiply quantified statement in predicate logic is reducible to a combination of these two standard forms.

(d) Interconversion of QL-1 and QL-2 forms of relations
in predicate logic (revised version of Section c)

(1) Generalization of the standard relations of this type
employed for the prenex normal form.

The following equations (77a,b,c,d) for implications are widely used in the construction of the prenex normal form in first order predicate logic, which is a universally valid form for a general statement in this system.

$$\begin{aligned} (\forall x)(\underline{a}x \implies \underline{b}) &\equiv (\exists x)(\underline{a}x) \quad \underline{b} ; \\ (\exists x)(\underline{a}x \implies \underline{b}) &\equiv (\forall x)(\underline{a}x) \end{aligned} \quad (77a,b)$$

$$\begin{aligned} (\forall x)(\underline{a} \implies \underline{b}x) &\equiv \underline{a} \quad (\forall x)(\underline{b}x) \\ (\exists x)(\underline{a} \implies \underline{b}x) &\equiv \underline{a} \quad (\exists x)(\underline{b}x) \end{aligned} \quad (77c,d)$$

These well-known formulae are proved for \underline{a} and \underline{b} having BA-1 truth values. When these relations were extended to BA-2 truth values, and QBA states for $q(\underline{i}x)$, it was found that a more general form of the relation had to be introduced, particularly when the implication is generalised to an SMS relation $\underline{Z}(k, \ell)$ of the types $\underline{A}(1, 1)$, $\underline{O}(1, 1)$, $\underline{I}(1, 1)$, $\underline{J}(1, 1)$, $\underline{E}(1, 1)$. It appears as if the most general form of the relation is as

given in (78a,b)

$$\begin{aligned} q(\underline{i}x) (s(kx)(\underline{ax}) \underline{Z}(1, 1) s(\underline{\ell})(\underline{6})) &\equiv q(\underline{i}x) (\underline{ax} \underline{Z}(k, \underline{\ell}) \underline{b}) \\ &\equiv q(\underline{i}'ax)(\underline{ax}) \underline{Z}' s(\underline{\ell}')(b) \end{aligned} \quad (78a)$$

$$\begin{aligned} q(\underline{j}y) (s(k)(\underline{a}) \underline{Z}(1, 1) s(\underline{\ell} y)(\underline{by})) &\equiv q(\underline{j}y) (\underline{a} \underline{Z}(k, \underline{\ell}) \underline{by}) \\ &\equiv s(k')(\underline{a}) \underline{Z}' q(\underline{j}'by)(\underline{by}) \end{aligned} \quad (78b)$$

In this, \underline{i}' can have the values \underline{i} , \underline{i}^c , \underline{i}^n or \underline{i}^ℓ , and similarly \underline{j}' can have the values \underline{j} , \underline{j}^c , \underline{j}^n or \underline{j}^ℓ . In the same way, k' can have the values k , k^n (and also k^c , k^ℓ) and ℓ' can have the values ℓ , ℓ^n (and also ℓ^c , ℓ^ℓ). In quantifier algebra, for \underline{i} corresponding to each one of the states $\forall, \exists, \wedge, \Phi$, \underline{i} , \underline{i}^c , \underline{i}^n and \underline{i}^ℓ lead to different permutations of the same four states. On the other hand, for the quantifier states $\zeta, \theta, \Delta, \Phi$, we have $q(\underline{i}) = q(\underline{i}^n)$ and $q(\underline{i}^c) = q(\underline{i}^\ell)$. In the same way, for the SNS states T and F, $k^n = k^c$ and the operation of negation (\underline{N}) and of complementation (\underline{M}), are equivalent in so far as they both lead to the interchange of the truth values T and F. However, for the states D and X, $k = k^n$ and $k^c = k^\ell$ and in particular, negation leaves D and X invariant while complementation interchanges them (see MR- regarding this theory). Although

SNS normally uses only negation (\neg), and all the logical equations of propositional calculus can be represented having only this negation operator (\neg) for vectors, we find there is a need for the use of both the operators \neg and \vee even on the SNS indices k and ℓ in connection with the equations/discussed in this section. We shall show how this happens, for instance, while applying the theory of (78a,b) to the quantifier states \sum and Δ . In fact, this will be extended to give a still more general formulation of the interconversion equations.

Actually, in the standard presentation of the prenex normal form, only implication is used. It can be shown that a general QL-1 form of relation, as in the l.h.sides of (78a or b), can be converted into an implication, as in (79a,b).

$$q(\underline{i}x)(\underline{a}x \underline{\sum} \underline{b}) \equiv q(\underline{i}'ax)(\underline{a}'x \longrightarrow \underline{b}') \quad (79a)$$

$$q(\underline{i}x)(\underline{a} \underline{\sum} \underline{b}x) \equiv q(\underline{i}'bx)(\underline{a}' \longrightarrow \underline{b}'x) \quad (79b)$$

For $\underline{\sum}(1, 1) = \underline{A}, \underline{O}, \underline{I}, \underline{J}$, the values of $\underline{i}', \underline{a}', \underline{b}'$ are as in Table 9 for both cases. The extension to $\underline{\sum}(k, \ell)$ is obvious.

Table 9. Conversion of a general QL-1 relation to an implication
as in (79a,b)

<u>Z</u>	<u>i'</u>	<u>a'</u>	<u>b'</u>
<u>A</u> :	<u>i</u> ⁿ	<u>a</u>	\neg <u>b</u>
<u>O</u> :	<u>i</u>	\neg <u>a</u>	<u>b</u>
<u>I</u> :	<u>i</u>	<u>a</u>	<u>b</u>
<u>J</u> :	<u>i</u>	\neg <u>b</u>	\neg <u>a</u>

(ii) Table of equivalent QL-1 and QL-2 forms of (78a,b)

We shall first consider Eq.(78a) and write the forward and backward transformations corresponding to it, from QL-1 to QL-2, and QL-2 to QL-1, as in (80a,b), for $\underline{Z} = \underline{Z}(1, 1) = \underline{A}, \underline{O}, \underline{I}, \underline{J}$ standing for the standard SNS connectives \wedge, \vee, \implies ,

$$\begin{aligned} \text{QL-2 to QL-1 : } q(\underline{i}ax)(\underline{ax}) \underline{Z} \underline{b} &\equiv q(\underline{i}'x)(s(k'))(\underline{ax} \underline{Z}' s(\ell'))(\underline{b}) \\ &\text{--- } \underline{Z}' \text{ limited to } \underline{Z} \text{ or } \underline{Z}^c \end{aligned} \quad (80a)$$

$$\begin{aligned} \text{QL-1 to QL-2 : } q(\underline{i}x) (s(ka)(\underline{ax}) \underline{Z} \underline{b}) &\equiv q(\underline{i}''ax)(\underline{ax}) \underline{Z}'' s(\ell'') \underline{b} , \\ &\text{--- } \underline{Z}'' \text{ limited to } \underline{Z} \text{ or } \underline{Z}^c \end{aligned} \quad (80b)$$

In the same way, the forward and backward transformations, from QL-1 to QL-2 and QL-2 to QL-1 of Eq.(78b) take the forms shown in (81a,b).

$$QL-2 \text{ to } QL-1 \quad \underline{a} \underline{z} \ q(\underline{j}by)(\underline{by}) \equiv q(\underline{j}'y)(s(k')(\underline{a}) \underline{z}' \ s(\ell')(\underline{by})) \quad (81a)$$

$$QL-1 \text{ to } QL-2 \quad q(\underline{j}y)(\underline{a} \underline{z} \ s(\ell)(\underline{by})) \equiv s(k'')\underline{a} \underline{z}'' \ q(\underline{j}''by)(\underline{by}) \quad (81b)$$

It will be seen that the structure of the two equations in (81) are identical with that of the two equations in (80), except that \underline{a} and \underline{b} are interchanged, x and y are interchanged, and the corresponding indices k and ℓ , and \underline{i} and \underline{j} , are also being interchanged. The relevant data for these two cases are given in Tables 10 and 11.

We have not given the corresponding tables for the four quantifier states $\underline{\Sigma}, \underline{\Theta}, \underline{\Delta}, \underline{\Phi}$, because there are some difficulties in formulating them in exactly the same manner as in Tables 10 and 11. We shall consider them after discussing the proof of the results for the four standard quantifier states, as given in Tables 10 and 11.

Before giving the proofs for the data given in Tables 10 and 11, it will be worthwhile to reduce the information given in these two tables to the bare minimum that is needed for working out the transformation between the QL-1 and QL-2 forms of a singly quantified elementary relation of the types

Table 10. Interconversion of QL-1 and QL-2 forms of quantified elementary relations — Transformation from the QL-2 to QL-1 forms: (a) For \underline{b} not quantified ; (b) For \underline{a} not quantified

(a) $\underline{a} \underline{Z} \underline{b} = \underline{q}'(\underline{a}' \underline{Z}' \underline{b}')$ (parameters as in (80a))				
$q(\underline{ia}x)$	$\underline{A}, \underline{O}, \underline{J}$		\underline{I}	
	$\underline{Z}' = \underline{Z}, \ell' = 1$ $q(\underline{i}'x) \quad k'$	$\underline{Z}' = \underline{Z}^C, \ell' = 1$ $q(\underline{i}'x) \quad k'$	$\underline{Z}' = \underline{Z}, \ell' = 1$ $q(\underline{i}'x) \quad k'$	$\underline{Z}' = \underline{Z}^C, \ell' = 1$ $q(\underline{i}'x) \quad k'$
$\forall (1)$	$\forall (i) \quad 1$	$\Phi (\underline{i}^n) \quad 1$	$\exists (\underline{i}^\ell) \quad 1$	$\Lambda (\underline{i}^m) \quad 1$
$\exists (6)$	$\exists (i) \quad 1$	$\Lambda (\underline{i}^n) \quad 1$	$\forall (\underline{i}^\ell) \quad 1$	$\Phi (\underline{i}^m) \quad 1$
$\Lambda (2)$	$\exists (i^n) \quad 2$	$\Lambda (\underline{i}) \quad 2$	$\forall (\underline{i}^m) \quad 2$	$\Phi (\underline{i}^\ell) \quad 2$
$\Phi (5)$	$\forall (i^n) \quad 2$	$\Phi (\underline{i}) \quad 2$	$\exists (\underline{i}^m) \quad 2$	$\Lambda (\underline{i}^\ell) \quad 2$
$q(\underline{jby})$	$q(j'y) \quad \ell'$ $\underline{Z}' = \underline{Z}, k' = 1$	$q(j'y) \quad \ell'$ $\underline{Z}' = \underline{Z}^C, k' = 1$	$q(j'y) \quad \ell'$ $\underline{Z}' = \underline{Z}^C, k' = 1$	$q(j'y) \quad \ell'$ $\underline{Z}' = \underline{Z}^C, k' = 1$
	$\underline{A}, \underline{O}, \underline{I}$		\underline{J}	
(b) $(\underline{a} \underline{Z} \underline{b}) = \underline{q}'(\underline{a}' \underline{Z}' \underline{b}')$ (parameters as in (81a))				

Table 11. Interconversion of QL-1 and QL-2 forms of quantified elementary relations — Transformation from the QL-1 to the QL-2 forms : (a) For \underline{b} not quantified ; (b) For \underline{a} not quantified

(a) $\underline{q}(\underline{a} \underline{Z} \underline{b}) = \underline{a}'' \underline{Z}'' \underline{b}''$ (parameters as in Eq.(80b))				
$q(ix)$	$\underline{A}, \quad \underline{Q}, \quad \underline{J}$		\underline{I}	
	$\underline{Z}'' = \underline{Z}, \ell'' = 1$ $q(\underline{j}''ax) \quad k$	$\underline{Z}'' = \underline{Z}^c, \ell'' = 1$ $q(\underline{j}''ax) \quad k$	$\underline{Z}'' = \underline{Z}, \ell'' = 1$ $q(\underline{j}''a''x) \quad k$	$\underline{Z}'' = \underline{Z}^c, \ell'' = 1$ $q(\underline{j}''a''x) \quad k$
$\forall (1)$	$\forall (\underline{j}) \quad 1$	$\Phi (\underline{j}^n) \quad 2$	$\exists (\underline{j}^\ell) \quad 1$	$\Lambda (\underline{j}^m) \quad 2$
$\exists (6)$	$\exists (\underline{j}) \quad 1$	$\Lambda (\underline{j}^n) \quad 2$	$\forall (\underline{j}^\ell) \quad 1$	$\Phi (\underline{j}^m) \quad 2$
$\Lambda (2)$	$\Phi (\underline{j}^\ell) \quad 2$	$\forall (\underline{j}^m) \quad 1$	$\Lambda (\underline{j}) \quad 2$	$\exists (\underline{j}^n) \quad 1$
$\Phi (5)$	$\Lambda (\underline{j}^\ell) \quad 2$	$\exists (\underline{j}^m) \quad 1$	$\Phi (\underline{j}) \quad 2$	$\forall (\underline{j}^n) \quad 1$
$q(jy)$	$q(\underline{j}''by) \quad \ell$ $\underline{Z}'' = \underline{Z}, k'' = 1$	$q(\underline{j}''by) \quad \ell$ $\underline{Z}'' = \underline{Z}^c, k'' = 1$	$q(\underline{j}''by) \quad \ell$ $\underline{Z}'' = \underline{Z}, k'' = 1$	$q(\underline{j}''by) \quad \ell$ $\underline{Z}'' = \underline{Z}^c, k'' = 1$
	$\underline{A}, \quad \underline{Q}, \quad \underline{I}$		\underline{J}	
(b) $\underline{q}(\underline{a} \underline{Z} \underline{b}) = \underline{a}'' \underline{Z}'' \underline{b}''$ (parameters as in Eq. (81b))				

considered for the two relevant cases. This is done by noting that the two columns in each half of Tables 10 and 11 are deducible, one from the other, by suitably interchanging $\underline{\underline{Z}}' = \underline{\underline{Z}}$ and $\underline{\underline{Z}}^c$ in the case of Table 10 and $k = 1$ and 2 in the case of Table 11. Consequently, we are able to reduce the general form of equations in (80) and (81) to the particular forms given given in Tables 12 and 13 for which the parameters required are tabulated in these two tables.

The most interesting thing that comes out of this analysis is that, if the input data are given with only the quantifiers \forall and \exists , but with the connective $\underline{\underline{Z}}$ having any one of the eight possibilities, namely $(\underline{\underline{A}}(k, \ell), \underline{\underline{O}}(k, \ell))$ or $(\underline{\underline{I}}(k, \ell), \underline{\underline{I}}^c(k, \ell))$, then the transformed relation in the other form is also expressible using the quantifiers \forall and \exists and having, for its connective $\underline{\underline{Z}}'$, or $\underline{\underline{Z}}''$, only one of the eight possibilities. This seems to be a good justification of the usual form of the expression of the "prenex normal form" in first order predicate calculus. We shall show how the special equations, with one term quantified and the other non-quantified, which is employed in Tables 10 to 13, can be extended to cover

(a) $q(\underline{i}x) (\underline{a}x \underline{z}(k, \ell) \underline{b}) \underline{b} = q(\underline{i}''ax) (\underline{a}x) \underline{z}''(k, \ell) \underline{b} ; \underline{z}'' = \underline{z} \sigma''$,

$\sigma'' = E \mapsto \underline{z}'' = \underline{z}, \sigma'' = M \mapsto \underline{z}'' = \underline{z}^c$

$q(\underline{i}x)$	σ''	$\underline{A}, \underline{O}, \underline{I}^c, \underline{J}$			$\underline{A}^c, \underline{O}^c, \underline{I}, \underline{J}^c$		
		$q(\underline{i}''ax)$	\underline{i}''	—	$q(\underline{i}''ax)$	\underline{i}''	—
\forall	E	\forall	\underline{i}	\underline{j}	\exists	\underline{i}^ℓ	\underline{j}^ℓ
\exists	E	\exists	\underline{i}	\underline{j}	\forall	\underline{i}^ℓ	\underline{j}^ℓ
\vee	M	\forall	\underline{i}^m	\underline{j}^m	\exists	\underline{i}^n	\underline{j}^n
\oplus	M	\exists	\underline{i}^m	\underline{j}^m	\forall	\underline{i}^n	\underline{j}^n

(b) $q(\underline{j}y) (\underline{a} \underline{z}(k, \ell) \underline{b}y) \underline{b} = q(\underline{j}''by) (\underline{b}y) \underline{z}''(k, \ell) \underline{b} ; \underline{z}'' = \underline{z} \sigma''$

$\sigma'' = E \mapsto \underline{z}'' = \underline{z}, \sigma'' = M \mapsto \underline{z}'' = \underline{z}^c$

$q(\underline{j}y)$	σ''	$q(\underline{j}''by)$	—	\underline{j}''	$\underline{A}^c, \underline{O}^c, \underline{I}^c, \underline{J}$		
		$\underline{A}, \underline{O}, \underline{I}, \underline{J}^c$			$\underline{A}^c, \underline{O}^c, \underline{I}^c, \underline{J}$		

an elementary relation containing two quantified terms
 with
 and $\underline{\underline{Z}}(k, \ell)$ being any one of the eight connectives mentioned
 above. However, there seems to be some question as to whether
 an elementary relation, in QL-1 or QL-2, of multiply quantified
 terms can be expressed completely in terms of \forall and \exists only.
 (See Section 5 of MR-53 for the general form of a multiply
 quantified term which requires all the possible seven permitted
 quantifier states in general. This should be carefully examined.)

(iii) Proof of the QL-1 — QL-2 transformation equations
for the standard quantifier states

We shall give the proof essentially for the data in
 Tables 10 and 11. All of them are derivable from just two
 equations corresponding to the first two rows in the second
 column of Table 10 — namely (82a,b) — and the corresponding
 ones in Table 11.

$$(\forall x)(\underline{\underline{a}}x) \underline{\underline{A}} \underline{\underline{b}} \equiv (\forall x)(\underline{\underline{a}}x \underline{\underline{A}} \underline{\underline{b}}) \quad (82a)$$

$$(\exists x)(\underline{\underline{a}}x) \underline{\underline{A}} \underline{\underline{b}} \equiv (\exists x)(\underline{\underline{a}}x \underline{\underline{A}} \underline{\underline{b}}) \quad (82b)$$

As will be seen from the third and fourth rows in the same
 column of Table 10, equations of the type of (82a,b) are not
 applicable for \wedge and $\bar{\Phi}$. The demonstration of this given below,

at the same time, brings out an essential difference between the quantifier generator states \forall and \exists . Thus, in (82a), $\underline{a}x$ is any term corresponding to the variable x , and it can be expressed equally well as the negation of another term, in the form $\underline{a}x = \neg \underline{a}_1x$. Making this substitution of $\neg \underline{a}x$ for $\underline{a}x$ in (82a), we obtain

$$(\exists x)(\underline{a}x) \underline{A} \underline{b} = (\forall x)(\neg \underline{a}x) \underline{A} \underline{b} = (\forall x)(\neg \underline{a}x \underline{A} \underline{b}) \quad (83a)$$

and similarly, from (82b), we obtain,

$$(\forall x)(\underline{a}x) \underline{A} \underline{b} = (\exists x)(\neg \underline{a}x \underline{A} \underline{b}) \quad (83b)$$

As is readily seen, the patterns of ^{the} equations in (82a,b), and in (83a,b), are not the same, in that k' is 1 for the former, and 2 for the latter. So also, $q(\underline{1}'x) = q(\underline{1}x)$ for the former, while $q(\underline{1}'x) = q(\underline{1}^n x)$ for the latter.

This points to the fact that the proof given in pages 59-65 (of Eqs (54a,b)) is questionable. In fact, the proof for $\underline{A}(1, 1)$ for $(\exists x)$ is not correct at all. It was only conjectured that the argument for $(\forall x)$ could be extended to $(\exists x)$ by interchanging T and F; but when attempts were made to complete that proof, they led to difficulties. Now, it has

become clear that the formula is different for $(\exists x)$, for both \underline{A} and \underline{Q} , from that for $(\forall x)$. So also, it has become clear that the formula for $\underline{E}(1, 1)$ has also to be reconsidered, since the transformed quantifier is different for \underline{I} and $\underline{J}(k, \ell)$. In both Tables 10 and 11, the two have quantifier states which are related by the operator \underline{L} (as e.g. (\forall, \exists) , (\wedge, \vee) etc). Consequently, it does not seem to be readily possible to obtain an equation of the type of (82), or (83), for \underline{E} , although this is possible for the four types of connectives \underline{A} , \underline{Q} , \underline{I} , \underline{J} .

Therefore, we give below a different proof for the parameters listed in Tables 10 and 11 for equations of the type (82) and (83). We shall illustrate the principle of the proof by means of the example of Eq.(82a). The l.h.s of (82a) has two input terms $\underline{a}x$ and \underline{b} , joined by the relation \underline{A} . We shall show that the truth value $(\underline{t}^{(2)})$ of this l.h.s in the QL-2 form, is the same as the truth value $(\underline{t}^{(1)})$ of the r.h.s giving the QL-1 form, for any given inputs $(\underline{a}'x, \underline{b}')$. For this purpose, it is sufficient to list the truth values $\underline{t}^{(1)}$ and $\underline{t}^{(2)}$ for the relative truth values $\underline{t}_1(\underline{g}_1 | \underline{a}')$ and $\underline{t}_2(T | \underline{b}') = \underline{t}(\underline{b}')$ having the possible SNS generator states T and F respectively. Thus,

we have four possibilities, namely (T, T), (T, F), (F, T), (F, F), and if we show that $\underline{t}^{(1)}$ and $\underline{t}^{(2)}$ are same for all the four, then the l.h.s and r.h.s are algebraically equivalent, and hence also logically equivalent.

Without going into further detail, we shall explain the entries in Table 14(a) which contains the proof of (82a).

Table 14(a). Proof of (82a) for the four possible combinations of truth values of \underline{a}' , \underline{b}'

1	2	3	4	5	6	7	8
$\underline{q}(\underline{a}')$	$\underline{t}_1(\underline{V} \underline{a}')$	$\underline{b}' = \underline{t}_2(\underline{b}')$	$\underline{t}_1 \underline{\wedge} \underline{t}_2$ $= \underline{t}^{(2)}$	$\underline{q}_s(\underline{b}')$	$\underline{q}(\underline{a}') \underline{\wedge} \underline{q}_s(\underline{b}')$ $= \underline{q}'$	$\underline{t}(\underline{V} \underline{q}') = \underline{t}^{(1)}$	$\underline{t}(\underline{\exists} \underline{q}') = \underline{t}''(1)$
\underline{V}	T	T	T	\underline{V}	\underline{V}	T	T
\underline{V}	T	F	F	$\underline{\Phi}$	$\underline{\Phi}$	F	F
$\underline{\wedge}$	F	T	F	\underline{V}	$\underline{\wedge}$	F	D
$\underline{\wedge}$	F	F	F	$\underline{\Phi}$	$\underline{\Phi}$	F	F

For mutually complementary quantifier states of $\underline{a}'x$, we employ \underline{V} and $\underline{\wedge}$ ($= \neg \underline{V}$), since this is the quantifier $\underline{q}(\underline{a}'x)$ of the relation on the l.h.s. This choice of $\underline{q}(\underline{a}'x)$ for calculating the relative truth value \underline{t}_1 as given below, has the

merit that \underline{t}_1 will have the values T, F as required in our test. Thus, \underline{t}_1 is given in column 2 corresponding to the choice of $\underline{q}(\underline{a}')$ and gives the relative truth value of \underline{V} for $\underline{a}'x$. The truth value \underline{t}_2 of \underline{b}' is taken to be also T and F, for each of the two possibilities of \underline{t}_1 . Consequently, we obtain for $\underline{t}^{(2)}$ the QL-2 truth value for each of the four possibilities and the results are given in column 4.

Coming to the r.h.s, we express both the terms $\underline{a}'x$ and \underline{b}' as QL-1 quantifier states. This can be done for \underline{b}' as $\underline{q}_s(\underline{b}')$, by taking the equivalent QBA states to be $\underline{V}, \underline{\Phi}$, corresponding to the SNS states T, F. Combining this with $\underline{q}(\underline{a}')$ in column 1, we obtain the QL-1 resultant of $\underline{q}(\underline{a}')$ and $\underline{q}_s(\underline{b}')$ for the binary forward relation with the connective \underline{A} , employing the QL-1A algebra discussed in Section 2(c). For this purpose, we can use the Table 3(a) of this report. (All the tables 3(a,b,c,d) will be used in the discussion hereafter, for QL-1A binary forward relations employing the connectives $\underline{A}, \underline{Q}, \underline{I}$ and \underline{E} .) This gives the data in column 6. By using the theory of QL-1 binary relations developed in Section 3 of this report, we can also obtain the relative truth value $\underline{t}(\underline{V}|\underline{q}')$ given in column 7, and this is equal to $\underline{t}(1)$, the truth value of the r.h.s of Eq.(82a). This choice of $\underline{V} = \underline{q}(\underline{i}'x)$

in $\underline{t}(q(\underline{1}'x) \mid q')$ is because this is the quantifier of the QL-1 relation in the r.h.s.

We have only to compare the two entries in all the four rows in column 4 and column 7, and it will be noticed that the two match, showing that the r.h.s and l.h.s of (82a) are equivalent. Just to show what happens if the wrong quantifier $q(\underline{1}'x)$ is used in the r.h.s, we give the relative truth value $\underline{t}'^{(1)}$ obtained by employing \exists instead of \forall , in column 8. It will be seen that, while three of the entries in columns 3 and 8 match, the ones in row 3 do not match, with one of them being F and the other D, showing that this particular quantifier for $q(\underline{1}'x)$ does not make the two sides equivalent.

Obviously, this proof is not simple to use for derivational purposes, but can be used for guessing at the correct solution. Hence, this procedure can always be used in an algorithmic manner for checking any transformation equation which is postulated. We shall indicate a more generalized form of this check below, and summarize it in Table 14(b) below for checking the correctness of Eq.(82b).

Table 14(b). Proof of (82b) for the three quantifier generator states of \underline{a}' and two SNS generator states of \underline{b}'

1	2	3	4	5	6	7	8
$\underline{q}(\underline{a}')$	$\underline{t}_1(\exists \underline{a}')$	$\underline{b}' = \underline{t}_2(\underline{b}')$	$\underline{t}_1 \underline{A} \underline{t}_2$ $= \underline{t}^{(2)}$	$\underline{q}_s(\underline{b}')$	$\underline{q}(\underline{a}') \underline{A} \underline{q}_s(\underline{b}')$ $= \underline{q}'$	$\underline{t}(\exists \underline{q}')$ $= \underline{t}^{(1)}$	$\underline{t}(\forall \underline{q}')$ $= \underline{t}'^{(1)}$
\forall	T	T	T	\forall	\forall	T	T
\forall	T	F	F	Φ	Φ	F	F
Σ	T	T	T	\forall	Σ	T	F
Σ	T	F	F	Φ	Φ	F	F
Φ	F	T	F	\forall	Φ	F	F
Φ	F	F	F	Φ	Φ	F	F

As given in the heading of Table 14(b), this more generalized approach makes definite use of the fact that any quantifier state can be represented in QBA as the Boolean sum of 1, 2 or 3 of the generator states \forall , Σ and Φ . Therefore, in making the check given in Table 14(a), it would be worthwhile making these as the possible quantifier states of $\underline{a}'x$, and combine them with the two SNS generator states T and F of \underline{b}' . The rest

of Table 14(b) is identical with that in Table 14(a), except for the changes required by the different equation, namely (82b), that is being tested. It will be seen that $\underline{t}^{(2)}$ in column 4 and $\underline{t}^{(1)}$ in column 7 agree for every one of the six rows, showing that this equation is correct. Just as in Table 14(a), a wrong choice of \forall for the quantifier on the r.h.s leads to the set of truth values in column 8 which agrees in seven cases, but disagrees for column 3, again showing that this quantifier will not fit the equation completely.

The above argument has been brought in mainly as an introduction to a still more general check that can be made of the QL-1 and QL-2 forms of an elementary relation in which both $\underline{a}x$ and $\underline{b}y$ are present as quantified terms, which will be discussed in the next section, when nine test values for the input pair $(\underline{a}'x, \underline{b}'y)$ should be used for checking for agreement between the QL-1 form and the QL-2 form of the same elementary relation.

It has been found that all the data in Tables 10 and 11 can be checked by this procedure. However, for derivational purpose, we require only the two equations (82a,b) and the

corresponding equations with non-quantified \underline{a} and quantified \underline{b} , and the others can be derived from these by procedures similar to those adopted for (83a,b). The details are omitted.

Before we pass on to the consideration of similar transformation equations between the QL-1 and QL-2 forms for the other four quantifier states, we shall first point out how the test fails to provide a positive answer to the check of equations of the type (84) below for the connective \underline{E} .

$$(\underline{q}x) (\underline{a}x) \underline{E} \underline{b} = (\underline{q}'x) (\underline{a}x \underline{E} \underline{b}), \underline{q}' = \underline{q} \quad (84)$$

We give below, in Table 14(c), the test following the same pattern as in Table 14(b). No further explanation is needed of the procedure adopted except to note that for calculating $\underline{t}^{(1)}$ in column 7, the Table 3(d) for the QL-1A relation \underline{E} is what is needed. It will be noted that columns 7 and 4 for $\underline{t}^{(1)}$ and $\underline{t}^{(2)}$ do not agree completely, in that in row 4, the two disagree, as pointed out by rings around the entries in the table. This discrepancy is of the same nature as the results obtained by using the wrong quantifier in Tables 14(a and b) and we have to conclude that Eq.(84) does not hold for $\underline{q} = \exists$. A similar check shows that it does not hold for any one of the four standard quantifiers $\forall, \exists, \wedge, \Phi$, for any \underline{q}' .

Table 14(c). Check of the relation $(\exists x)(\underline{a}x) \underline{\underline{E}} \underline{b} \equiv (\exists x)(\underline{a}x \underline{\underline{E}} \underline{b})$
for six combinations of truth values.

1	2	3	4	5	6	7
$\underline{q}(\underline{a}')$	$\underline{t}_1(\exists \underline{a}')$	$\underline{b}' = \underline{t}_2(\underline{b}')$	$\underline{t}_1 \underline{\underline{E}} \underline{t}_2$ $= \underline{t}^{(2)}$	$\underline{g}_s(\underline{b}')$	$\underline{q}(\underline{a}') \underline{\underline{E}} \underline{g}_s(\underline{b}')$ $= \underline{q}'$	$\underline{t}(\exists \underline{q}')$ $= \underline{t}^{(1)}$
V	T	T	T	V	V	T
V	T	F	F	Φ	Φ	F
Σ	T	T	T	V	Σ	T
Σ	T	F	(F)	Φ	Σ	(T)*
Φ	F	T	F	V	Φ	F
Φ	F	F	T	Φ	V	T

A very simple explanation of why this happens can be indicated for this example, checked in Table 14(c), by noting the well-known fact that $\underline{a} \underline{\underline{E}} \underline{b} = (\underline{a} \underline{\underline{I}} \underline{b}) \wedge (\underline{a} \underline{\underline{J}} \underline{b})$. Consequently, the l.h.s of (84) for $\underline{q} = \exists$ is equivalent to

$$((\exists x)(\underline{a}x) \underline{\underline{I}} \underline{b}) \wedge ((\exists x)(\underline{a}x) \underline{\underline{J}} \underline{b}) \quad (85a)$$

$$\equiv ((\forall x)(\underline{a}x \underline{\underline{I}} \underline{b})) \wedge ((\exists x)(\underline{a}x \underline{\underline{J}} \underline{b})), \text{ from Table 10 } (85b)$$

* Since $\underline{t}^{(1)} = \underline{t}^{(2)}$ for all rows, except row 4, for which $\underline{t}^{(2)} \rightarrow \underline{t}^{(1)}$, we can conclude that l.h.s of (84) implies r.h.s of (84), for $\underline{q} = \underline{q}' = \exists$ (see subsection (iv) below).

Since the quantifiers for the two terms in (85b) are not the same, it does not follow that their conjunction is equal to $(\exists x)(\underline{a}x \underline{E} \underline{b})$. In fact, in the test demonstrated in Table 14(c), the difference between $\underline{t}^{(1)}$ and $\underline{t}^{(2)}$ arises essentially for the combination $(\underline{\sum}, F)$ of (ax, \underline{b}) . It is interesting to note that the quantifier for which this test disagrees is $\underline{\sum}$, which is also the difference between \forall and \exists , which are the quantifiers of the two terms in the r.h.s of (85b) — $(\exists = \forall \oplus \underline{\sum})$.

In fact, the proof given in Section 6(c)(i), for the connective $\underline{E}(1, 1)$ for $(\forall x)$, has the defect that the two cases examined are only those for which $\underline{a}'x = (\forall x)(ax)$ and $(\underline{\Phi} x)(ax)$, and the proof has not been written corresponding to the middle two rows of Table 14(c), namely those involving $(\underline{\sum} x)(ax)$. A very simple demonstration that the Eq.(86) given below does not hold for $(\underline{\sum} x)(\underline{a}'x)$ can be given as follows:

$$(\underline{\sum} x)(\underline{a}'x) \underline{E} \underline{b} \equiv (\underline{\sum} x)(\underline{a}'x \underline{E} \underline{b}) \quad (86)$$

We must remember the fact that, in the QL-2 logical relation on the l.h.s, $(\underline{\sum} x)(\underline{a}'x)$ is collective in its interpretation

which means that $\underline{a}'x$ is true for some x , but not for all x . Consequently, if $\underline{b} = F$, it means that $\underline{a}'x$ is either true for all x or is false for all x . This means that the relation is true for (Θx) — i.e. "for all x " or "for no x ", collectively.

On the other hand, the quantifier $(\exists x)$ on the r.h.s has an individual application — i.e for every x . Consequently, the predicate $(\underline{a}'x \underline{E} \underline{b})$ is true for $(\exists x)$ — i.e. for some x , but not for all x . If \underline{b} is F , this means that the predicate is false for some x , but not for all x . However, since in general, $(\exists x)(\underline{t}x) \equiv (\exists x)(\neg \underline{t}x)$, the r.h.s is true only for the range $(\exists x)$ of the variable x . Thus, we get the conclusion that the l.h.s is true for $(\Theta x) \equiv \neg(\exists x)$, while the r.h.s is true for $(\exists x) \equiv \neg(\Theta x)$. In other words, while the l.h.s is F , the r.h.s is T , for $\underline{b} = F$. This is precisely what is shown in row 4 of Table 14(c). Thus, we have obtained a simple proof without Boolean algebra, but merely from the description of $(\exists x)$ as "there exists, but not for all" ($\equiv \exists \& \forall$) and the very existence of this state for quantified logic requires that the transformation Eq.(86) is not true for both $\underline{b} = T$ and $\underline{b} = F$ and is not an equivalence relation between its l.h.s and r.h.s.

An interesting consequence of this to the standard formulation of predicate logic is discussed in the next subsection (iv). As will be shown therein, this formulation which employs BA-1 truth values for the predicate, and has only four quantifier states $\forall, \exists, \wedge, \Phi$, is incomplete, and leads, in fact, even to inconsistent results for the QL-1 to QL-2 transformation equations.

(iv) A new inconsistency theorem for the standard formulation of first order predicate logic.*

In Section 7(d)(v) of MR-53, it was shown, in Table 7(d), that the QL-1A product $\sum \underline{E} \Theta$ is equal to \sum from the definition of \sum as a QBA state. However, the definition of the two states \sum and Θ in terms of the four standard quantifier states, leads to two different values — \sum , or Δ , — for $\sum \underline{E} \Theta$, according as the operation of Boolean product to obtain \sum from $\sum = \exists \otimes \wedge$ is carried out first, and that of the Boolean sum to obtain $\Theta = \forall \oplus \Phi$ is carried out thereafter, or these are carried out in the reverse order. There seems to be no indication from standard theory as to which of these is the correct one. On the other hand, the definition of \sum as a

*This section is left as it is, although the inconsistency can be explained away by careful analysis, which is given in the next section

generator state, and of Θ as the sum of the generator states

\forall and \exists , gives a unique answer (Σ) for $\Sigma \in \Theta$, and it is the same as one of the two possibilities mentioned above which are obtained by the above procedure. Other similar inconsistencies have been pointed out in Section 5(c)(iii) of this report, when the standard approach of employing only the four quantifier states $\forall, \exists, \wedge, \phi$, with the latter two defined as the complements of the former two, is adopted for the calculations. Now, in the previous subsection (iii), we have observed that for the QL-1, 2 transformation equations employing the quantifier (Σx) for ($\underline{q} x$)(\underline{ax}) $\underline{E} \underline{b}$ also, we obtain discrepancies if only BA-1 truth values are employed, and only the standard quantifier states are used for obtaining Σ and Θ in terms of them. Thus, the proof in Section 6(c)(i) holds for

$$(\forall x)(\underline{ax}) \underline{E} \underline{b} = (\forall x)(\underline{ax} \underline{E} \underline{b}), \text{ for } \underline{q} = \underline{q}' = \forall \text{ in (84)} \quad (87a)$$

and it can readily be extended to give (87b).

$$(\exists x)(\underline{ax}) \underline{E} \underline{b} = (\exists x)(\underline{ax} \underline{E} \underline{b}), \text{ for } \underline{q} = \underline{q}' = \exists \text{ in (84)} \quad (87b)$$

This proof is essentially based on BA-1 truth values. However, the equations (85a) and (85b) indicate that Eq.(84) does not

hold for either $\underline{q} = \underline{q}' = \forall$, or \exists , so that both (87a) and (87b) cannot be valid. This is clearly a contradiction, since the same formula is provable to be true by one line of argument and provable to be false by another line, indicating inconsistency in the whole procedure.

However, we believe that this is only due to the inadequacy in the formulation, ^{of} employing only BA-1 truth values and using only two quantifier states \forall and \exists as basic for the definition of all quantifiers. As mentioned in MR-52, this is insufficient for a proper definition of all possible quantifiers, and if we extend the formalism by employing BA-2 truth values, and also employ three quantifier states \forall , \exists , Φ as generator states, then both lines of argument do lead to the result that Eq.(86) is not satisfied, and consequently, we come to the conclusion that Eq.(84), for the connectives $\underline{E}(1, 1)$, and $\underline{E}(1, 2)$, are not satisfiable for any $\underline{q} = \underline{q}'$.

This matter requires further consideration and this will be done in a future report. However, a brief outline of a ^{much simpler} possible proof, of the inconsistency involving the connective \underline{E} in QL-1, 2 transformation equations obtained by using standard procedures, may be given as follows, without requiring the definition of the new quantifier states Σ and Θ .

We start with Eqs (88a,b) which are proved in all books on logic (See e.g. Eqs.(7), (10), p.118 of Howard DeLong, "A Profile of Mathematical Logic, Addison Wesley, 1970).

$$((\forall x)(\underline{a}x) \implies \underline{b}) \equiv (\exists x)(\underline{a}x \implies \underline{b}) \quad (88a)$$

$$(\underline{b} \implies (\forall x)(\underline{a}x)) \equiv (\forall x)(\underline{b} \implies \underline{a}x) \quad (88b)$$

Consequently, we have

$$((\forall x)(\underline{a}x) \iff \underline{b}) \equiv ((\forall x)(\underline{a}x) \implies \underline{b}) \wedge (\underline{b} \implies (\forall x)(\underline{a}x)) \quad (89a)$$

$$\equiv (\exists x)(\underline{a}x \implies \underline{b}) \wedge (\forall x)(\underline{b} \implies \underline{a}x) \quad (89b)$$

$$\not\equiv (\forall x)(\underline{a}x \iff \underline{b}) \quad (89c)$$

However, we have proved in Section 6(c)* (pages 61-62) that

$$(\forall x)(\underline{a}x) \underline{\equiv} \underline{b} \equiv (\forall x)(\underline{a}x \underline{\equiv} \underline{b}) \quad (90)$$

By showing that, for both $\underline{b} = T$ and $\underline{b} = F$, the l.h.s implies the r.h.s and the r.h.s implies the l.h.s. Clearly, Eqs. (89) and (90) are mutually contradictory, and both have been derived using standard predicate calculus without the need for either SNS algebra or QBA algebra for quantifiers.

This seems to be a very simple and straightforward demonstration of the inadequacy of BA-1 truth values for propositional calculus and predicate calculus.

*This proof is invalid and therefore the conclusion in the last para above is not substantiated (see next section).

(v) Re-examination of QL-1,2 transformation equations for
connective \underline{E}

In this section, we shall give the derivation of the correct relational equations between QL-2 and QL-1 forms given in the l.h.s and r.h.s of Eq.(84) for \underline{q} and $\underline{q}' = \underline{\vee}$ and $\underline{\exists}$. It will be shown that four such relations are derivable from well-known equivalence relations involving implications, and they will be substantiated by means of the truth value checks of the type indicated in Tables 14(a,b,c). This will indicate the great value of the truth value check, employing/QBA generator states for quantifiers and the SNS generator states T and F for non-quantified terms. The following equivalence relations (91) to (94) are well-known and are found in every book on logic (see e.g. DeLong: "A Profile of Mathematical Logic" p. 118 Eqs 7-10).

$$((\underline{\vee} x)(\underline{a}x) \Rightarrow \underline{b}) \equiv (\underline{\exists} x)(\underline{a}x \Rightarrow \underline{b}) \quad (91)$$

$$((\underline{\exists} x)(\underline{a}x) \Rightarrow \underline{b}) \equiv (\underline{\vee} x)(\underline{a}x \Rightarrow \underline{b}) \quad (92)$$

$$(\underline{a} \Rightarrow (\underline{\vee} x)(\underline{b}x)) \equiv (\underline{\vee} x)(\underline{a} \Rightarrow \underline{b}x) \quad (93)$$

$$(\underline{a} \Rightarrow (\underline{\exists} x)(\underline{b}x)) \equiv (\underline{\exists} x)(\underline{a} \Rightarrow \underline{b}x) \quad (94)$$

We wish to find the nature of the connective which relates the l.h.s and r.h.s in the four equations (95-98) below between

a QL-2 equivalence relation and a QL-1 equivalence relation of the same type as those in Eqs (91-94):

$$((\forall x)(\underline{ax}) \equiv \underline{b}) \underline{\underline{Z}}_1 (\forall x)(\underline{ax} \equiv \underline{b}) \quad (95)$$

$$((\forall x)(\underline{ax}) \equiv \underline{b}) \underline{\underline{Z}}_2 (\exists x)(\underline{ax} \equiv \underline{b}) \quad (96)$$

$$((\exists x)(\underline{ax}) \equiv \underline{b}) \underline{\underline{Z}}_3 (\forall x)(\underline{ax} \equiv \underline{b}) \quad (97)$$

$$((\exists x)(\underline{ax}) \equiv \underline{b}) \underline{\underline{Z}}_4 (\exists x)(\underline{ax} \equiv \underline{b}) \quad (98)$$

It is possible to find out the nature of the SNS connectives

$\underline{\underline{Z}}_1$ to $\underline{\underline{Z}}_4$ by suitable manipulation of Eqs (91-94), provided

use is made of the well-known implication relation ⁽⁹⁹⁾ between

$(\forall x)(\underline{px})$ and $(\exists x)(\underline{px})$, where the predicate \underline{px} can be a simple term by itself, or can be a relation of the type $\underline{ax} \implies \underline{b}$ or

\underline{ax} , etc. :

$$(\forall x)(\underline{px}) \implies (\exists x)(\underline{px}) \quad (99)$$

Eqs. (100a,b) and (101a,b) follow from (99):

$$(\underline{b} \implies (\forall x)(\underline{ax})) \implies (\underline{b} \implies (\exists x)(\underline{ax})) \quad (100a)$$

$$((\exists x)(\underline{ax}) \implies \underline{b}) \implies ((\forall x)(\underline{ax}) \implies \underline{b}) \quad (100b)$$

$$(\forall x)(\underline{ax} \implies \underline{b}) \quad (\exists x)(\underline{ax} \implies \underline{b}) \quad (101a)$$

$$(\forall x)(\underline{b} \implies \underline{ax}) \quad (\exists x)(\underline{b} \implies \underline{ax}) \quad (101b)$$

Using (91-94) and (100a,b),(101a,b), we can obtain the nature of the connectives $\underline{Z}_1, \underline{Z}_2, \underline{Z}_3, \underline{Z}_4$ as in (102) — details omitted,

$$\begin{aligned} \underline{Z}_1 &= \underline{J} = \underline{I}(2,2) & ; & \quad \underline{Z}_2 = \underline{I} = \underline{I}(1,1) & ; \\ \underline{Z}_3 &= \underline{J} = \underline{I}(2,2) & ; & \quad \underline{Z}_4 = \underline{I} = \underline{I}(1,1) \end{aligned} \quad (102a,b,c,d)$$

however,
We shall, / indicate the great convenience of the 3×2 truth table of the type contained in Tables 14(a,b,c) for checking these relations. The corresponding tables for Eq.(84) with $\underline{q}, \underline{q}' = \underline{\forall}, \underline{\exists}$ are given in Tables 15(a) and (b). In Table 15(a), the l.h.s is taken to be $((\underline{\forall} x)(\underline{ax} \equiv \underline{b}))$ and the r.h.s equal to $(\underline{\forall} x)(\underline{ax} \equiv \underline{b})$ and $(\underline{\exists} x)(\underline{ax} \equiv \underline{b})$. In Table 15(b), the l.h.s is taken to be $((\underline{\exists} x)(\underline{ax} \equiv \underline{b}))$, while the r.h.s has the two cases the same as for 15(a). The layout of the table follows essentially the pattern of Table 14(c) and is not described. The results are discussed in relation to Eqs. (95-98) and (102a,b,c,d) after the tables.

We have only to compare the entries in columns 4 and 7 to get the right connective \underline{Z}_1 for Eq.(95). It will be noticed that all the rows except row 4 have entries which agree, showing that the implication can be both ways, while in row 4

Table 15(a). Check of Equations (95), (96)

1	2	3	4	5	6	7 ⁺	8 ⁺
$\underline{q}(\underline{a}')$	$\underline{t}_1(\underline{V} \underline{a}')$	$\underline{b}' = \underline{t}_2(\underline{b})$	$\underline{t}_1 \stackrel{E}{=} \underline{t}_2$ $= \underline{t}^{(2)}$	$\underline{q}_s(\underline{b}')$	$\underline{q}(\underline{a}') \stackrel{E}{=} \underline{q}_s(\underline{b}')$ $= \underline{q}'$	$\underline{t}(\underline{V} \underline{q}')$ $= \underline{t}^{(1)}_{\underline{V}}$	$\underline{t}(\underline{\exists} \underline{q}')$ $= \underline{t}^{(1)}_{\underline{\exists}}$
\underline{V}	T	T	T	\underline{V}	\underline{V}	T	T
\underline{V}	T	F	F	$\underline{\Phi}$	$\underline{\Phi}$	F	F
$\underline{\Sigma}$	F	T	(F)	\underline{V}	$\underline{\Sigma}$	F	(T)
$\underline{\Sigma}$	F	F	(T)	$\underline{\Phi}$	$\underline{\Sigma}$	(F)	T
$\underline{\Phi}$	F	T	F	\underline{V}	$\underline{\Phi}$	F	F
$\underline{\Phi}$	F	F	T	$\underline{\Phi}$	\underline{V}	T	T

+7 for (95); 8 for (96)

(\Leftarrow) (\Rightarrow)

Table 15(b). Check of Equations (97), (98)

1	2	3	4	5	6	7 ⁺	8 ⁺
$\underline{q}(\underline{a}')$	$\underline{t}_1(\underline{\exists} \underline{a}')$	$\underline{b}' = \underline{t}_2(\underline{b})$	$\underline{t}_1 \stackrel{E}{=} \underline{t}_2$ $= \underline{t}^{(2)}$	$\underline{q}_s(\underline{b}')$	$\underline{q}(\underline{a}') \stackrel{E}{=} \underline{q}_s(\underline{b}')$ $= \underline{q}'$	$\underline{t}(\underline{V} \underline{q}')$ $= \underline{t}^{(1)}_{\underline{V}}$	$\underline{t}(\underline{\exists} \underline{q}')$ $= \underline{t}^{(1)}_{\underline{\exists}}$
\underline{V}	T	T	T	\underline{V}	\underline{V}	T	T
\underline{V}	T	F	F	$\underline{\Phi}$	$\underline{\Phi}$	F	F
$\underline{\Sigma}$	T	T	(T)	\underline{V}	$\underline{\Sigma}$	(F)	T
$\underline{\Sigma}$	T	F	(F)	$\underline{\Phi}$	$\underline{\Sigma}$	F	(T)
$\underline{\Phi}$	F	T	F	\underline{V}	$\underline{\Phi}$	F	F
$\underline{\Phi}$	F	F	T	$\underline{\Phi}$	\underline{V}	T	T

+ 7 for (97) ; 8 for (98)

(\Leftarrow) (\Rightarrow)

alone $\underline{t}^{(2)} = T$ while $\underline{t}^{(1)} = F$. This means that the implication is from QL-1 to QL-2. Since the direction is from QL-2 form to QL-1 in (95), the implication is $\underline{J} \equiv \underline{I}(2,2)$, i.e. "reverse imply", in (95).

On the other hand, on comparing columns 4 and 8, all the rows agree except row 3, for which $\underline{t}^{(2)}$ is F and $\underline{t}^{(1)}$ is T. Consequently, the implication is from QL-2 to QL-1 form and therefore \underline{Z}_2 for Eq.(96) is $\underline{I} = \underline{I}(1,1)$, i.e "imply".

In the same way, in Table 15(b), on comparing columns 4 and 7, we find that only the entries in row 3 differ and $\underline{t}^{(2)} = T$ while $\underline{t}^{(1)} = F$ so that the implication is from right to left, indicating that \underline{Z}_3 for Eq.(97) is J, i.e. "reverse imply". Similarly on comparing columns 4 and 8, we find that the only difference is in the entries for row 4, for which, however, the implication is from left to right, so that \underline{Z}_4 for (98) is \underline{I} , i.e. "imply".

Thus, by very simply listing the truth table and its consequences, we have been able to deduce what are the relevant connectives in Eqs (95-98). The results in Tables 15(a) and 15(b)

can be summarized as in (103),

$$\begin{array}{ccc}
 (\forall x)(\underline{a}x \equiv \underline{b}) & & \\
 \swarrow & & \searrow \\
 (\forall x)(\underline{a}x) \equiv \underline{b} & (\exists x)(\underline{a}x) \equiv \underline{b} & (103) \\
 \swarrow & & \searrow \\
 (\exists x)(\underline{a}x \equiv \underline{b}) & &
 \end{array}$$

and they agree with the deductions made in (102), thus showing consistency between the standard formulation and our QBA representation of quantifiers.

We shall now proceed to consider QL-1,2 transformation relations for elementary statements in which both \underline{a} and \underline{b} are quantified. To do this quite generally, we need, in addition to formulae given in Tables 12 and 13 for the four standard quantifiers $\forall, \exists, \wedge, \Phi$, also those for ζ and Θ . However, the latter two lead to some peculiarities in the transformation equations which are better appreciated after the general case is discussed for the standard quantifiers. Therefore, the general QL-1 - QL-2 interconversion formulae are discussed in the next subsection (e), after which we shall come back to such equations for the quantifiers ζ and Θ .

— QL-2 interconversion of statements in general

We are now in a position to reconsider the discussion in Sections 6(c)(ii) and 6(c)(iii), which are to be replaced by this Section 6(e). The general form of the interconversion equations when one of the terms \underline{a} and \underline{b} is quantified and the other is non-quantified, is shown in (104a,b) and (105a,b) below and they are readily capable of being combined to give the corresponding equations for both \underline{a} and \underline{b} being quantified, which is the one that is needed essentially for converting a general statement in first order predicate logic into the so-called prenex normal form. In fact, in our method of implementation of QL-1 and QL-2 equations, we do the reverse — namely of converting the prenex normal form into a sequence of QL-1 or QL-2 elementary relations and implementing them thereafter. (See Section 7.)

$$q(\underline{i}ax)(\underline{ax}) \underline{\underline{Z}} \underline{b} = q(\underline{i}'ax)(\underline{ax} \underline{\underline{Z}}' \underline{b}) ; \quad (104a)$$

$$\underline{a} \underline{\underline{Z}} q(\underline{j}by)(\underline{by}) = q(\underline{j}'by)(\underline{a} \underline{\underline{Z}}' \underline{by}) \quad (104b)$$

$$q(\underline{i}x)(\underline{ax} \underline{\underline{Z}} \underline{b}) = q(\underline{i}''ax)(\underline{ax}) \underline{\underline{Z}}'' \underline{b} \quad (105a)$$

$$q(\underline{j}y)(\underline{a} \underline{\underline{Z}} \underline{by}) = \underline{ax} \underline{\underline{Z}}'' q(\underline{j}''by)(\underline{by}) \quad (105b)$$

The data for Eqs.(104) and (105) correspond to Tables 12 and 13 respectively and the parameters on the r.h.s of these equations are obtainable from the contents of the tables.

Before we consider how the above equations can be combined, we shall mention the necessary conditions for them to be valid. For (104a), \underline{b} must be free of x so that the truth value of \underline{b} does not depend in any way on the truth value of \underline{ax} , and, vice-versa, for (104b), \underline{a} must be free of y . Consequently, it is quite possible to have \underline{b} quantified for y as $q(\underline{jby})(\underline{by})$ in (104b), and \underline{a} quantified for x as $q(\underline{iax})(\underline{ax})$ in (104a). Similar considerations hold for (105a) and (105b). Consequently, we obtain the following general form of the transformation equations for an elementary relation in predicate logic between two quantified terms. (These equations (106a,b) are quite different from (73a,b) which should be replaced by these.) In (106a,b) \underline{Z} stands for $\underline{A}(k, \ell)$, $\underline{O}(k, \ell)$, $\underline{I}(k, \ell)$ or $\underline{J}(k, \ell)$ as the case may be. In the forward direction from QL-2 to QL-1 we have (106a),

$$q(\underline{iax})(\underline{ax}) \underline{Z}(k, \ell) q(\underline{jby})(\underline{by}) \equiv q(\underline{i'ax}) q(\underline{j'by}) (\underline{ax} \underline{Z}(k', \ell') \underline{by}) \quad (106a)$$

and in the reverse direction from QL-1 to QL-2 form we have (106b).

$$q(\underline{iax}) q(\underline{jby}) (\underline{ax} \underline{Z}(k, \ell) \underline{by}) \equiv q(\underline{i''ax})(\underline{ax}) \underline{Z}''(k'', \ell'') q(\underline{j''by})(\underline{by}) \quad (106b)$$

(i) Implementation of the transformation from the QL-2 to the QL-1 form

of (106a)

As already mentioned, this implementation is done by combining the information contained in Tables 12(a) and (b) for the two individual terms $\underline{a}x$ and $\underline{b}y$ contained in (104a,b). This is done by carrying out the operation in (104a) first, and then that in (104b). Thus, the former leads to $q(\underline{i}'ax)$ and $\underline{Z}_1(k', \ell)$ from $q(\underline{i}ax)$ and $\underline{Z}(k, \ell)$. When the second equation in (104b) is applied, we obtain $q(\underline{j}'by)$ and $\underline{Z}'(k', \ell')$ from $q(\underline{j}by)$ and $\underline{Z}_1(k', \ell)$. Thus, by individually applying the two processes described in Tables 12(a) and (b), we obtain $q(\underline{i}'ax)$, $q(\underline{j}'by)$ and $\underline{Z}'(k', \ell')$ in (106a). In this particular transformation from QL-2 to QL-1 form, it so turns out that the two equations are independently applicable; and the calculation of k' does not affect ℓ or $q(\underline{j}by)$ and the calculation of ℓ' does not affect k or $q(\underline{i}ax)$. Further, the nature of the connective \underline{Z} (namely \underline{A} , \underline{O} , \underline{I} or \underline{J}) is not modified by the transformation. Therefore, the whole procedure is consistent and this is further checked by the fact that the reverse transformation from QL-1 to QL-2 described in the next subsection (ii) leads back to the original equation or its equivalent.

(ii) Implementation of the transformation from the QL-1
the QL-2 form

Exactly as in the previous case, we now apply Tables 13(a) and (b) to Eq.(106b) via Eqs (105a,b). In this case, we do have the possibility of the connective $\underline{\underline{Z}}$ also changing as the process of first transferring the quantifier for $\underline{\underline{ax}}$ from the QL-1 to the QL-2 form is carried out, before that for the second term by is done. However, the two are sequentially implementable, and it is only necessary to convert the SNS connective $\underline{\underline{Z}}^c(k, \ell)$ into its equivalent form $\underline{\underline{Z}}^d(k^c, \ell^c)$ where $\underline{\underline{Z}}^d$ stands for the Le Morgan associate of $\underline{\underline{Z}}$. Thus, $\underline{\underline{O}}^c(1,1) = \underline{\underline{A}}(2,2)$, $\underline{\underline{A}}^c(1,2) = \underline{\underline{O}}(2,1)$, etc., so that $\underline{\underline{A}}^d = \underline{\underline{O}}$ and $\underline{\underline{O}}^d = \underline{\underline{A}}$. We are not giving formulae for $\underline{\underline{I}}$, $\underline{\underline{J}}$, $\underline{\underline{I}}^c$ and $\underline{\underline{J}}^c$ since they can always be converted/in the / of either $\underline{\underline{O}}(k, \ell)$ or $\underline{\underline{A}}(k, \ell)$. Therefore, in this discussion, we shall restrict ourselves to $\underline{\underline{A}}(k, \ell)$ and $\underline{\underline{O}}(k, \ell)$ for the connectives.

The procedure for performing the transformation in (106b) is as follows. First the quantifier $q(\underline{\underline{jby}})$ is transferred inside the bracket as $q(\underline{\underline{j"by}})$ with a corresponding change of $\underline{\underline{Z}}(k, \ell)$ to $\underline{\underline{Z}}_1(k, \ell)$ (where $\underline{\underline{Z}}_1 = \underline{\underline{Z}}$ or $\underline{\underline{Z}}^c$) by using Eq. (105b). We then obtain as an intermediate form,

$$q(\underline{i}ax) (\underline{ax} \underline{Z}_1(k, \ell) q(\underline{j}by)(\underline{by})), \quad \underline{Z}_1(k, \ell) = \underline{Z}(k, \ell) \text{ , or } \underline{Z}^d(k^c, \ell^c),$$

according as $\sigma' = E \text{ or } M \quad (107)$

Then, Eq.(105a) is applied to remove the bracket and convert (107) to the QL-2 form when we obtain $q(\underline{i}ax)$ and $\underline{Z}''(k, \ell)$ also, from $\underline{Z}_1(k, \ell)$.

Although the above procedure looks complicated in description, it is actually very simple, and we shall illustrate it by two examples after pointing out that the full tables 12 and 13 are not necessary and that the upper halves of each table, ^{for} the quantifiers \forall and \exists , are sufficient. However, it is absolutely necessary to have all the eight connective operators $\underline{A}(k, \ell)$ and $\underline{O}(k, \ell)$ for $k, \ell = 1, 2$. When this simplification is made, we effectively find that the operation of σ' in Table 12 and σ'' in Table 13 becomes no longer necessary, as both have the value ^E for the first two rows in the body of the two tables. Consequently, the connective $\underline{Z}(k, \ell)$ of the l.h.s and the r.h.s are the same for both (106a) and (106b), and only one of the two quantifiers \forall or \exists occurs for the r.h.s. also. This agrees very well with the standard procedure for writing the prenex normal form.

(iii) Examples of interconversion formulae

We shall give one example each of (106a) and (106b).

As already mentioned, it will be preferable to take the normal form of the input in the l.h.s to have only the quantifier states \forall and \exists . This is always possible for both the QL-1 form and the QL-2 form as indicated below. If the input quantifier $q(\underline{i}ax)$ or $q(\underline{j}by)$ is \wedge or Φ , then we change it to $q(\underline{i}^m ax)$ or $q(\underline{j}^m by)$, as the case may be, as shown below. Thus for the QL-2 form

$$q(\underline{i}ax)(\underline{ax}) \underline{Z}(k, \ell) q(\underline{j}by)(\underline{by}) \equiv q(\underline{i}^m ax)(\underline{ax}) \underline{Z}(k^c, \ell) q(\underline{j}by)(\underline{by}),$$

$$q(\underline{i}^m) = \neg q(\underline{i}) \quad (108a)$$

$$q(\underline{i}ax)(\underline{ax}) \underline{Z}(k, \ell) q(\underline{j}by)(\underline{by})$$

$$\equiv q(\underline{i}ax)(\underline{ax}) \underline{Z}(k, \ell^c) q(\underline{j}^m by)(\underline{by}), q(\underline{j}^m) = \neg q(\underline{j}) \quad (108b)$$

Consequently, if either the term \underline{ax} in (108a) or the term \underline{by} in (108b) has \wedge or Φ as quantifier, it is converted into \forall or \exists with only the connective operator/changed/into any one of the eight possible connective operators. This can be done even if both $q(\underline{i}ax)$ and $q(\underline{j}by)$ are of the type \wedge or Φ .

For QL-1, the procedure is slightly different and is based essentially on the formulae

$$q(\underline{\exists}x)(\underline{p}x) = q(\underline{\exists}^n \underline{p}x)(\neg \underline{p}x) ; \quad (109a)$$

and

$$\neg q(\underline{\exists}^{\ell} \underline{p}x)(\underline{p}x) = q(\underline{\exists}^{\ell} \underline{p}x)(\neg \underline{p}x) \quad (109b)$$

We shall not formally write the general formula for converting a QL-1 elementary relation into the normal form, but indicate it by examples below.

The specific examples for the two cases are as follows.

Normalization of QL-2 form: We shall give two or three examples of this to indicate how (108a,b) are applied.

$$(\underline{\Lambda}x)(\underline{a}x) \underline{Q}(2,1) (\underline{\forall}y)(\underline{b}y) = (\underline{\forall}x)(\underline{a}x) \underline{Q}(1,1) (\underline{\forall}y)(\underline{b}y) \quad (110a)$$

$$(\underline{\exists}x)(\underline{a}x) \underline{Q}(1,1) (\underline{\Phi}y)(\underline{b}y) = (\underline{\exists}x)(\underline{a}x) \underline{Q}(1,2) (\underline{\exists}y)(\underline{b}y) \quad (110b)$$

$$(\underline{\Phi}x)(\underline{a}x) \underline{A}(1,1) (\underline{\Lambda}y)(\underline{b}y) = (\underline{\exists}x)(\underline{a}x) \underline{A}(2,2) (\underline{\forall}y)(\underline{b}y) \quad (110c)$$

It will be noticed that all that is done is to convert $\underline{\Lambda}$ or $\underline{\Phi}$ into its complement $\underline{\forall}$ or $\underline{\exists}$ and change the corresponding k or ℓ in $\underline{Z}(k, \ell)$ into its complement. Thus, in (110a) only the first term has $\underline{\Lambda}$ changed to $\underline{\forall}$ and $\underline{Q}(2,1)$ changed to $\underline{Q}(1,1)$ and similarly in (110b), only the second term has $\underline{\Phi}$ changed into $\underline{\exists}$ and $\underline{Q}(1,1)$ correspondingly changed to $\underline{Q}(1,2)$. On the other hand,

in (110c), since the ^{two} terms have $(\bar{\Phi} x)$ and $(\bar{\Lambda} y)$ respectively, they are changed into $(\exists x)$ and $(\forall y)$, and the connective $\underline{\underline{A}}(1,1)$ has both k and ℓ changed to their complements, to yield $\underline{\underline{A}}(2,2)$ on the r.h.s.

Normalization of QL-1 form: This is slightly more complicated, as it depends on the sequence of the two quantifiers.

$$(\forall x)(\bar{\Phi} y) (\underline{\underline{a}}x \underline{\underline{A}}(1,1) \underline{\underline{b}}y) = (\forall x)(\forall y)(\underline{\underline{a}}x \underline{\underline{Q}}(2,2) \underline{\underline{b}}y) \quad (111a)$$

$$\begin{aligned} (\bar{\Lambda} x)(\bar{\Phi} y) (\underline{\underline{a}}x \underline{\underline{Q}}(2,1) \underline{\underline{b}}y) &= (\exists x) \left[\neg(\bar{\Phi} y) (\underline{\underline{a}}x \underline{\underline{Q}}(2,1) \underline{\underline{b}}y) \right] \\ &= (\exists x)(\exists y)(\underline{\underline{a}}x \underline{\underline{Q}}(2,1) \underline{\underline{b}}y) \end{aligned} \quad (111b)$$

$$\begin{aligned} (\bar{\Phi} x)(\exists y) (\underline{\underline{a}}x \underline{\underline{Q}}(1,1) \underline{\underline{b}}y) &= (\forall x) \left[(\bar{\Phi} y) (\underline{\underline{a}}x \underline{\underline{Q}}(1,1) \underline{\underline{b}}y) \right] \\ &= (\forall x)(\forall y) (\underline{\underline{a}}x \underline{\underline{A}}(2,2) \underline{\underline{b}}y) \end{aligned} \quad (111c)$$

The procedure is to carry out the change from $(\bar{\Lambda}, \bar{\Phi})$ to (\forall, \exists) first for $\underline{\underline{q}}x$ and then for $\underline{\underline{q}}y$ sequentially. This agrees with the well-known convention regarding the scope of application of a quantifier to the expression that comes to the right of it. Thus, in (111a), $(\forall x)$ need not be changed. Therefore, only $(\bar{\Phi} y)$ is changed to $(\forall y)$ and correspondingly the expression forming the predicate within the bracket is converted into its negation involving the replacement of the operator $\underline{\underline{Z}}(k, \ell)$ by

$\equiv^d(k^c, \ell^c)$. In (11b), on the other hand, only the first quantifier $(\bigwedge x)$ has to be changed, which becomes $(\exists x)$ on the r.h.s. Then, the negation of the expression to the right is the complementation of the quantified relation, which can be expressed by taking the complement of the quantifier, with the predicate unchanged, as in the first line of (111b). In the second line, we replace $\neg(\Phi y)$ by $(\exists y)$, and since this is a quantifier of the normal form, there is no change in the predicate and we obtain the expression in the second line of (111b).

If both of them have to be normalised, then qx is changed first, and then qy , as in (111b). Thus, (Φx) is first made into $(\forall x)$, when $(\exists y)$ becomes its complement (Φy) , as in the first line of (111c). However, (Φy) has once again to be changed to the normal form, which is done by negating the predicate, and converting (Φy) into $(\forall y)$, as in the second line of (111c).

The above three examples indicate the general rules for changing the quantifier in a QL-1 statement into the normal form for x alone, and for y alone, and for both x and y .

If $\underline{q}y$ is the one that is to be normalized, only the predicate is converted into its negation as in (111a).

If both $\underline{q}x$ and $\underline{q}y$ are to be normalized, then $\underline{q}x$ is negated into $\underline{q}^n x$ and $\underline{q}y$ is complemented to $\underline{q}^m y$ and the predicate is unaffected. On the other hand, if $\underline{q}x$ alone is to be normalized, then $\underline{q}x$ is negated to $\underline{q}^n x$ and, at the same time $\underline{q}y$ is converted to $\underline{q}^{\ell} y$, with simultaneous negation of the relation in the predicate. In all cases, the negation of the predicate is obtained by complementing the SNS operator $\underline{z}(k, \ell)$ to become $\underline{z}^d(k^c, \ell^c)$.

Concluding remarks (added June 4, 1987)

This report had to be discontinued for various reasons and it is therefore closed at this stage. The QL-1 and QL-2 forms of quantifier logic are briefly discussed in four lectures which form MR-55 and MR-56. The formulae developed therein have been converted into computer programs for single statements in QL-1 and QL-2 formalism, and these are contained in ALOG-56 (MATLOG Part II). In view of these later developments, this report is to be considered only as a step in the development of the BVMF theory of quantifiers.

BOOLEAN VECTOR-MATRIX FORMULATION (BVMF) OF LOGIC

Lecture 1 and 2 , Series 2

Lecture 1 : Boolean algebraic formalism (BA-2) for
propositional calculus

Lecture 2 : Logical graphs and their implementation

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

BOOLEAN VECTOR-MATRIX FORMULATION (BVMF) OF LOGIC

Lectures 1 and 2 , Series 2

Lecture 1 Boolean algebraic formalism (BA-2) for
propositional calculus

Lecture 2 : Logical graphs and their implementation

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

PREFACE

The two lectures whose contents are presented in this report were given in the Lecture Series-2, but they have been superseded in many ways in the Lecture Series-3. Therefore, they have not been very carefully checked for errors or omissions, but have been left in the form in which they were presented. They, as well as Lectures 3 and 4 in the next report MR-56, are consistent with the formalism and notation developed in Matphil Reports 52, 53 and 54.

Lecture-1 is written in the form of a complete report, while Lecture-2 contains a good part of the text in pages 1-26, while the essential formulae, tables, and figures, are given in the appended overhead projector sheets A1 to A15. Cross references between the two are given where essential, but has not been made exhaustive.

CONTENTS

Page No.

Lecture 1 Boolean algebraic formalism (BA-2) for propositional calculus

Scope of the talks	i-vii
Special features of BVMF	viii
1. Boolean representation (BA-1) of classical logic	1
2. States and connective operators in SNS	2
(a) States	2
(b) Matrix operators	3
(c) General 2x2 matrix operators in SNS	4
(i) Unary relation	4a
(ii) Binary relation	5
(iii) Connection between the unary form and binary form of a relation	9
(d) Boolean operators in SNS	10
3. Logical graphs and their implementation	13
(a) Elements of logical graphs	13
(b) Simple example of a logical argument	15
(i) Application of vidya operator	15
(c) Reversal of argument from contradictions	18
(d) Proof that any argument can be implemented as a sequence of elementary relations	20
(e) Implementation of circular arguments	23
(i) Discussion for $\underline{Z} = \underline{I}$	24
(ii) Discussion for $\underline{Z} = \underline{E}$	25

Boolean Vector-Matrix Formulation of Logic

Scope of the talks

The purpose of these talks is to give a bird's eye view of the studies made on the application of Boolean algebra systematically for the implementation of logical relations in general, — in sentential calculus, in quantified predicate calculus, and in higher order logic, including multiple valued logic. All of them can be given a unified representation in the Boolean vector-matrix formulation (BVMF). First, we shall give a general picture of the formalism for propositional calculus, in which the standard employment of BA-1 algebra, with only two elements 1 and 0 standing for T and F, is extended to BA-2 algebra employing Boolean 2-vectors $(a_\alpha \ a_\beta)$ with $a_\alpha \ a_\beta = 1, 0$, and it leads to the four possibilities $T = (1 \ 0)$, $F = (0 \ 1)$, $D = (1 \ 1)$, $X = (0 \ 0)$ for the truth values. The introduction of the two new truth values D(doubtful) and X(impossible), in addition to the standard T and F, makes the algebra very much more amenable for implementation.

The BVMF theory is extended to the general theory of relations between two sets A and B having m and n members, in which a_i is related to b_j by the matrix R_{ij} , with $R_{ij} = 1$ if a_i is related to b_j , and 0 if they are unrelated. In this, the idea that any term a_i or b_j can only have the states T and F is maintained (although the use of the four states T, F, D, X is also possible and is discussed.)

From the general theory of relations, it is possible to give a definition for the quantifier states \forall and \exists , in terms of well-formed formulae in propositional calculus. However, they are found to obey a higher order Boolean algebra (BA-3), than the Boolean algebra BA-2 of propositional calculus. A systematic presentation of quantifier logic for first order predicate calculus can be given in terms of this formalism involving BA-3 algebra, with BA-2 truth values. Practically all formulae in standard predicate logic is reproduceable in this algebra, which has the great merit that it is algorithmic and is readily computerizable.

Both for propositional calculus and predicate logic, it is possible to represent a logical relation between two

terms joined by a connective such as "and", "or", "equ", "not", "if" etc., by a binary vector-matrix relation of the type $a_i R_{ij} b_j$, in which a_i is an m -vector and b_j is an n -vector ($m = n = 2$ for propositional calculus, $m = n = 3$ for quantifier states). In its implementation such a relation can lead to two forms of equations, termed respectively unary and binary, which have the forms

Unary relation

$$\sum_i a_i R_{ij} = b_j \quad (1)$$

Binary relation

$$\sum_i \sum_j a_i R_{ij} b_j = c_\alpha, \quad \sum_i \sum_j a_i R_{ij}^c b_j = c_\beta, \quad \underline{c} = (c_\alpha, c_\beta) \quad (2)$$

In (2), R_{ij}^c is the complement of the matrix R_{ij} , with each matrix element being complemented. It is interesting that Eq.(2) leads necessarily to a 2-vector \underline{c} , having the four truth values of BA-2, which is employed in our algebra for propositional calculus. Since the idea of a doubtful state is fundamental to our theory, this algebra is given the name Syad Nyaya System (syad = may be, nyaya = logic, in Sanskrit) (SNS for short).

Both in the SNS algebra of PC and BA-3 quantifier algebra, all implementable equations in logic have the pattern of either Eq.(1) or Eq.(2), and these equations have the property of being reversible. Thus, in Eq.(1) a_i can be given from b_j by the transpose R^t of the relation R .

$$\sum b_j R_{ij}^t = a_i ; R_{ij}^t = R_{ji} \quad (3)$$

and Eq.(2) can be reversed given c and a_i , the details of which will be given in Section . It turns out that this binary reverse automatically leads to the unary relation.

This concept of reversibility of a statement in logic (corresponding to the inversion of a linear equation in linear algebra by matrix inversion) is a novel feature of the theory and is extremely useful for back-tracking an argument from a contradiction to the exact origin of the contradiction.

So also, the check/^{for}contradiction is automatic, and is made by a new operator called "vidya" (Knowledge in Sanskrit), which yields the state $X = (0\ 0)$ if the same term has contradictory inputs having the truth values T and F. The introduction of this operator vidya (V) and of the related operator U for unanimity is again a novel feature. In fact, they are the

generalization in BA-n of the representation of the logical AND and logical OR in BA-1 algebra. The distinction between the logical AND and OR as normally understood, which are representable by matrices $\begin{matrix} A_{and} & 0 \\ 1 & \end{matrix}$, and their counterpart for checking purposes as \underline{V} and \underline{U} , is again a novel feature of the whole algebra, which makes the writing of programs very convenient.

Apart from the theory of implementation of the logic of atomic statements, the analysis of an argument in terms of its logical graph and its translation into BVMF can also be given in very general terms. In fact, all possible types of statements, with or without quantifiers, which form an argument, can be broken down into elementary statements of the type of Eqs (1) or (2) (and their generalizations), and then arranged in sequential order for successive implementation. This theory ^{has been} worked out particularly for propositional calculus, and, interestingly, the essential basis in terms of graph theory of this algorithm is completely applicable also to predicate logic, except that the terms become quantified in the latter case.

The Boolean vector-matrix formalism arose in our studies as the consequence of attempts at simulating the logical equations in SNS and QL by electronic circuitry. Some of the circuits are interesting and they will be briefly considered in the last lecture along with algorithms for the implementation of BVMF equations. It is proposed to give six lectures on topics connected with the titles given below.

- 1 Principles of BA-2 algebra of SNS with four truth values T, F, D, X.
- 2 Logical graphs and their implementation in propositional calculus.
- 3 Theory of relations in BVMF and application to information processing.
- 4 BA-3 algebra of BVMF for quantifiers — essential principles.
- 5 BVMF theory of first order predicate calculus.
- 6 Computer algorithms and circuits for implementing BVMF formulae.

They will contain the outline of the theory and formulae that are necessary for working out problems or proofs in

sentential logic, multivalued logic of the theory of relations and in quantified predicate logic. Only the formulae are given and no proofs are supplied. Also there is no attempt made to make the set of formulae given complete, the idea being that the essential background of our new approach be indicated and illustrated.

We shall use the term CL to indicate "classical logic", standing for the BA-1 theory of propositional calculus, and the term SNS to indicate the BA-2 theory of propositional calculus. All terms in PC(SNS) are indicated by lower case letters which are double underlined (a), and all connectives by capital letters, double underlined, e.g (A, O,...). The truth values of SNS are indicated by T, F, D, X. The term QL (standing for quantifier logic) is used for first order predicate logic and higher order logic employing quantifiers, in general. Terms in quantifier logic are denoted by curly underlines as in ax, by etc., and connectives in predicate logic are indicated by z and z in QL-1 and QL-2 (the distinction will be explained in due course). Some special features of BVMF are pointed out in the next page viii.

Special Features of Boolean Vector Matrix Formalism

- 1 A very general and unified presentation of logical relations.
- 2 Introduction of Boolean 2-vectors $(a_\alpha \ a_\beta)$ in BA-2 algebra for terms and the new truth values $D = T \vee F$, $X = \neg T \wedge \neg F$ in addition to T and F, and 2x2 matrices for connectives.
- 3 Distinction between $\underline{a} \wedge \underline{b} = \underline{c}$ and $\underline{a}_1 \wedge \underline{a}_2 = \underline{a}$ by different operators \underline{A} and \underline{V} .
- 4 Implementation of relations and their universal reversibility, as in

$$(x, ax = y) \mapsto y \equiv (y, y/a = x) \mapsto x \quad : \text{Unary relation}$$

$$(x, y, xy = c) \mapsto c \equiv (x, c, c/x = y) \mapsto y \quad : \text{Binary relation}$$
- 5 Definition of quantifiers in terms of logical equations (well-formed formulae) in propositional calculus (PC), for the formulation of QL (predicate calculus), via BA-3.
- 6 Representation of arguments in PC and QL by logical graphs, and their reversal from contradiction to trace back the source for it.
- 7 Theory of relations between n-vector a_i and m-vector b_j via matrix R_{ij} as in

$$\sum_i a_i R_{ij} = b_j, \quad i = 1 \text{ to } m, \quad j = 1 \text{ to } n$$
 and generalization to tensor relations $R_{ij} \dots \ell$.
- 8 Practically all known theorem in PC and QL can be treated in BVMF, as well as information processing via the theory of relations (in BA-1 and BA-2).
- 9 New types of electronic circuits for representing BVMF equations.

Lecture 1 Boolean algebraic formalism (BA-2) for propositional calculus and the theory of logical graphs

This report briefly summarizes the formulae which we believe are necessary for working out any problem, or proof, in sentential logic in particular which form the basis for the multivalued logic in the theory of relations in general, and the theory of logical graphs for representing an argument and its implementation for practical problems. Only the formulae are given and no proofs are supplied. For convenience different symbols are used for standard classical logic using two truth values T and F and for SNS formulae requiring four truth values T, F, D, X. The former are denoted by non-underlined symbols (e.g. a, b) and the latter by double underlined symbols (e.g. a, b, 0, A, Z etc).

1. Boolean representation (BA-1) of classical logic

$$(a) \quad \underline{\text{States}} \quad T \mapsto 1, \quad F \mapsto 0 \quad (1.1)$$

(b) Operations

$$\underline{\text{Negation}} \quad \neg a = b \mapsto a^c = b \quad (1^c \mapsto 0, 0^c \mapsto 1) \quad (1.2)$$

$$\underline{\text{Conjunction}} \quad a \wedge b = c \mapsto a \otimes b = c \quad (1.3)$$

$$\underline{\text{Disjunction}} \quad a \vee b = c \mapsto a \oplus b = c \quad (1.4)$$

Combination of negation and either conjunction or disjunction is illustrated by the example of

$$\underline{\text{Implication}} \quad a \Rightarrow b \equiv \neg a \vee b \mapsto a^c \oplus b \quad (1.5)$$

.2.

The three basic operations in (1.2), (1.3) and (1.4) are sufficient for constructing the Boolean representation in BA-1 of any sentential logic statement, and the fact that BA-1 is a ring, is a proof of the completeness of the logic of sentential calculus.

We also note that (1.2), (1.3) and (1.4) have truth tables in the standard form and that these truth tables are sufficient to work out the state of truth ($a = 1$ or 0) for any term a defined by a set of logical equations in sentential logic.

2. States and connective operators in SNS

(a) States:

$$T \mapsto (1 \ 1) ; F \mapsto (0 \ 1) ; D \mapsto (1 \ 1), X \mapsto (0 \ 0) \quad (2.1a)$$

To distinguish from BA-1 representation, the state vectors in SNS are denoted by double underlined symbols such as $\underline{\underline{a}}$, $\underline{\underline{b}}$ etc. The symbol $\underline{\underline{a}}$ stands for the vector $(a_\alpha \ a_\beta)$ where a_α and a_β denote the two Boolean components of the state vector. The properties of D and X are given below in (2.1b).

$$D = (1 \ 1) = (1 \ 0) \oplus (0 \ 1) = T \vee F ; X = (1 \ 0) \otimes (0 \ 1) = T \wedge F \quad (2.1b)$$

.3.

The symbols \oplus and \otimes in SNS is defined later (Section 2(d)) for SNS and BA-2, and in Section 3 for BA-n. The operator c (complement) yields in SNS Eq.(2,2).

$$(\underline{a}^c = \underline{b}) \equiv a_{\alpha}^c = b_{\alpha} , a_{\beta}^c = b_{\beta} , \text{ so that } T^c = F, F^c = T, D^c = X, X^c = D \quad (2.2)$$

This operator is different from "negation" which is represented by the matrix operator \underline{N} (see Section 2(c) for details).

(b) Matrix operators

In SNS we have two types of operators which we denote by the name "matrix" and "Boolean". The analogues of Eqs (1.2), (1.3) and (1.4) are matrix operators in SNS and are defined first.

$$\underline{\text{Negation:}} \quad \underline{a} \underline{N} = \underline{b} \mapsto \langle a | N | = \langle b | , \quad | N | = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (2.3a)$$

Note that

$$\langle a | N | N | = \langle a | E | = \langle a | , \quad | E | = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (|E| = \text{equivalence}) \quad (2.3b)$$

Conjunction

$$\underline{a} \underline{A} \underline{b} = \underline{c} \mapsto \langle a | A | b \rangle = c_{\alpha} , \langle a | A^c | b \rangle = c_{\beta} , \langle c | = (c_{\alpha} \ c_{\beta}) \quad (2.4a)$$

$$| A | = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Disjunction

$$\underline{a} \underline{O} \underline{b} = \underline{c} \mapsto \langle a | O | b \rangle = c_{\alpha} , \langle a | O^c | b \rangle = c_{\beta} , \langle c | = (c_{\alpha} \ c_{\beta}) \quad (2.4b)$$

$$| O | = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

The Dirac bracket notation for the matrix connective \underline{Z} is defined as follows.

$$\langle a | Z | = \langle b | : \bigoplus_{\lambda} a_{\lambda} \otimes z_{\lambda\mu} = b_{\mu} ; \lambda, \mu = \alpha, \beta \quad (2.5a)$$

$$\langle a | Z | b \rangle = c : \bigoplus_{\lambda} \bigoplus_{\mu} a_{\lambda} \otimes z_{\lambda\mu} \otimes b_{\mu} = c ; \lambda, \mu = \alpha, \beta \quad (2.5b)$$

$$\langle a | b \rangle = c : \bigoplus_{\lambda} a_{\lambda} \otimes b_{\lambda} = c ; \lambda = \alpha, \beta \quad (2.6)$$

In essence, Eqs (2.3 - 2.6) are sufficient to construct all formulae needed for SNS logic, except for Boolean operators discussed below. However, in order to make the matrix calculus complete and applicable, we list these formulae under (c).

(c) General 2x2 matrix operators in SNS

Define

$$|Z| = \begin{pmatrix} z_{\alpha\alpha} & z_{\alpha\beta} \\ z_{\beta\alpha} & z_{\beta\beta} \end{pmatrix}, z_{\lambda\mu} = 0 \text{ or } 1 \text{ and } |Z^c| = \begin{pmatrix} z_{\alpha\alpha}^c & z_{\alpha\beta}^c \\ z_{\beta\alpha}^c & z_{\beta\beta}^c \end{pmatrix} \quad (2.7a,b)$$

Then, two types of relations, which we have named unary relation and binary relation can be defined as below. In (2.7b) Z^c is termed the complement of Z . Similarly, the complement of a vector $\langle a |$ can be defined as in (2.7c)

$$\langle a^c | = (a_{\alpha}^c \ a_{\beta}^c) ; \text{ Also } \langle a | M | = \langle a^c | \quad (2.7c)$$

.4a.

The operator M is not a matrix operator, but a Boolean operator as described in Section 2(d), but it is given the format of a matrix operator for convenience.

(i) Unary relation: This can be defined in the forward direction by (2.8a) and the same relation in the backward direction as in (2.8b)

$$\underline{a} \underline{Z} = \underline{b} \mapsto \langle \underline{a} | \underline{Z} | = \langle \underline{b} | \quad (2.8a)$$

$$\underline{b} \underline{Z}^t = \underline{a} \mapsto \langle \underline{b} | \underline{Z}^t | = \langle \underline{a} | \quad (2.8b)$$

In this the transpose \underline{Z}^t of the matrix operator \underline{Z} is defined by (2.8c)

$$Z_{\lambda\mu}^t = Z_{\mu\lambda} \quad (2.8c)$$

Note that (2.8a) and (2.8b) are equivalent. This will be particularly clear from Section 3 on the theory of relations.

Examples are :

$$\langle \underline{a} | \underline{E} | = \langle \underline{b} | , \quad \langle \underline{a} | \underline{N} | \equiv \langle \underline{a} | \underline{E}^c | = \underline{b} , \quad \langle \underline{a} | \underline{I} | = \langle \underline{b} | \quad (2.9a,b,c)$$

which stand for $\underline{a} \underline{E} = \underline{b} , \underline{a} \underline{N} \equiv \underline{a} \underline{E}^c = \underline{b} , \underline{a} \underline{I} = \underline{b} .$ If we want

to represent a relation like $\neg \underline{a} \Rightarrow \neg \underline{b}$ we do it as in (2.9d)

$$\underline{a} \underline{N} \underline{I} \underline{N} = \underline{b} \mapsto \langle \underline{a} | \underline{N} | \underline{I} | \underline{N} | = \langle \underline{b} | \quad (2.9d)$$

(ii) Binary relations: Binary relations of the type (2.5)

and (2.6) can be generalized to the form

$$\underline{a} \underline{Z} \underline{b} = \underline{c} \mapsto \langle a | Z | b \rangle = c_\alpha, \langle a | Z^c | b \rangle = c_\beta, (c_\alpha \ c_\beta) = \langle c | \quad (2.10)$$

In (2.10), $|Z|$ can be any one of the 16 possible 2x2 Boolean matrices. However, only ten of these are of direct interest to the commonly used logical connectives. These are listed in Table 1. From the particular examples given in Table 1, it is immediately verified that the matrix $|Z|$ has its four components $Z_{\lambda\mu}$ the same as the Boolean elements in the truth table of the connective \underline{Z} in classical 2-valued logic. This isomorphism between the matrix algebra of SNS for binary relations and the truth table method of classical logic establishes the complete concordance of the two for all formulae represented in terms of the states T and F only. It can be shown that, if $\langle a |$ and $\langle b |$ have only the state (1 0) and (0 1), then the vector $\langle c |$ of Eq.(2.10) also has only these two states T and F.

Although we shall primarily use the symbols $\neg, \wedge, \vee, \Rightarrow, \equiv$ for BA-1 algebra, and the symbols $\underline{Z} = \underline{N}, \underline{A}, \underline{O}, \underline{I}, \underline{E}$ for BA-2 algebra, wherever it leads to simplicity, the standard

Table 1The Ten Connectives of Propositional Calculus
and their BA-2 matrices in SNS

<u>A</u> (and)	<u>N</u> <u>A</u> (not implicate <u>J</u> ^c)	<u>A</u> <u>N</u> (not imply <u>I</u> ^c)	<u>N</u> <u>A</u> <u>N</u> (nor <u>O</u> ^c)
<u>a</u> <u>∧</u> <u>b</u> = <u>c</u>	<u>¬a</u> <u>∧</u> <u>b</u> = <u>c</u>	<u>a</u> <u>∧</u> <u>¬b</u> = <u>c</u>	<u>¬a</u> <u>∧</u> <u>¬b</u> = <u>c</u>
<u>a</u> <u>A</u> <u>b</u> = <u>c</u>	<u>a</u> <u>N</u> <u>A</u> <u>b</u> = <u>c</u>	<u>a</u> <u>A</u> <u>N</u> <u>b</u> = <u>c</u>	<u>a</u> <u>N</u> <u>A</u> <u>N</u> <u>b</u> = <u>c</u>
$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \underline{A}(1,1)$	$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \underline{A}(2,1)$	$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \underline{A}(1,2)$	$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \underline{A}(2,2)$
<u>O</u> (or)	<u>N</u> <u>O</u> (imply = <u>I</u>)	<u>O</u> <u>N</u> (implicate = <u>J</u>)	<u>N</u> <u>O</u> <u>N</u> (nand, <u>A</u> ^c)
<u>a</u> <u>∨</u> <u>b</u> = <u>c</u>	<u>¬a</u> <u>∨</u> <u>b</u> = <u>c</u>	<u>a</u> <u>∨</u> <u>¬b</u> = <u>c</u>	<u>¬a</u> <u>∨</u> <u>¬b</u> = <u>c</u>
<u>a</u> <u>O</u> <u>b</u> = <u>c</u>	<u>a</u> <u>N</u> <u>O</u> <u>b</u> = <u>c</u>	<u>a</u> <u>O</u> <u>N</u> <u>b</u> = <u>c</u>	<u>a</u> <u>N</u> <u>O</u> <u>N</u> <u>b</u> = <u>c</u>
$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \underline{O}(1,1)$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \underline{O}(2,1)$	$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \underline{O}(1,2)$	$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = \underline{O}(2,2)$
<u>E</u> (Equivalent)	<u>N</u> (Negation) = <u>E</u> ^c	<u>I</u> = <u>N</u> <u>O</u> (imply) (suff)	
(<u>a</u> <u>≡</u> <u>b</u>) = <u>c</u>	(<u>a</u> <u>≠</u> <u>b</u>) = <u>c</u>	<u>J</u> = <u>O</u> <u>N</u> (implicate*) (nec.)	
<u>a</u> <u>E</u> <u>b</u> = <u>c</u>	<u>a</u> <u>N</u> <u>b</u> = <u>c</u>	<u>I</u> ^t = <u>J</u> = <u>N</u> <u>I</u> <u>N</u> (contrapositive form)	
$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \underline{E}(1,1)$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \underline{E}(2,1)$	$(\underline{b} \Rightarrow \underline{a}) \equiv (\neg \underline{a}) \Rightarrow (\neg \underline{b})$	

De Morgan relations : A^c = N O N ; O^c = N A N*The name "reverse imply" is also used, and represented by a ← b

.7.

symbols for CL will be utilized for BA-2 and BA-3 formulae also. Table 1 gives a summary of the ten connectives of propositional calculus, along with the corresponding matrices, and the description of each connective in terms of \underline{A} , \underline{O} or \underline{E} .

Very recently, we have introduced a notation for the four states of truth (truth values) T, F, D and X by denoting them by $s(k)$, $k = 1, 2, 3, 4$ respectively. Thus

$$\begin{aligned} s(1) &= T = \begin{pmatrix} 1 & 0 \end{pmatrix} ; & s(2) &= F = \begin{pmatrix} 0 & 1 \end{pmatrix} ; \\ s(3) &= D = \begin{pmatrix} 1 & 1 \end{pmatrix} ; & s(4) &= X = \begin{pmatrix} 0 & 0 \end{pmatrix} \end{aligned} \quad (2.11a)$$

Using a notation analogous to a quantifier, we write \underline{a} and $\neg \underline{a}$ as $s(1)(\underline{a})$ and $s(2)(\underline{a})$ respectively. Then, a general connective \underline{Z} , among the ten in Table 1, can be given a notation $\underline{Z}(k, \ell)$ as follows:

$$\begin{aligned} \underline{a} \underline{Z} \underline{b} &\equiv \underline{a} \underline{Z}(1,1) \underline{b}, & \neg \underline{a} \underline{Z} \underline{b} &= \underline{a} \underline{Z}(2,1) \underline{b} \\ \underline{a} \underline{Z} \neg \underline{b} &= \underline{a} \underline{Z}(1,2) \underline{b}, & \neg \underline{a} \underline{Z} \neg \underline{b} &= \underline{a} \underline{Z}(2,2) \underline{b} \end{aligned} \quad (2.11b)$$

This notation is also incorporated in Table 1. Consequently, a general binary relation in SNS is expressible as $\underline{a} \underline{Z}(k, \ell) \underline{b}$ and its consequences ^{are} / that the terms \underline{a} and \underline{b} have the sign

.8.

(affirmation or negation) according as k and $\ell = 1$ or 2 respectively. It will be noticed that \underline{a} and \underline{b} are only dummy symbols indicating the name of the term concerned, but their nature, i.e. $s(k)$ and $s(\ell)$, is contained in the symbol for the connective operator $\underline{Z}(k, \ell)$. So also \underline{Z} stands for the type of connective (namely \wedge , \vee or \equiv) that is involved, and the four possibilities for the former two cases, and two in the third case, are distinguished by the indices (k, ℓ) .

Equation (2.10) for a binary relation can be put in an even simpler form by employing BA-1 algebra. Thus, for $\underline{a} \underline{Z} \underline{b} = \underline{c}$, we have the following:

$$\text{For } \underline{Z} = \underline{A}, \quad a_\alpha \wedge b_\alpha = c_\alpha; \quad a_\beta \vee b_\beta = c_\beta \quad (\wedge \equiv \text{AND}, \otimes) \quad (2.12a)$$

$$\text{For } \underline{Z} = \underline{O}, \quad a_\alpha \vee b_\alpha = c_\alpha; \quad a_\beta \wedge b_\beta = c_\beta \quad (\vee \equiv \text{OR}, \oplus) \quad (2.12b)$$

$$\text{For } \underline{Z} = \underline{E}, \quad a_\alpha \equiv b_\alpha = c_\alpha; \quad a_\beta \equiv b_\beta = c_\beta \quad (\equiv \equiv \text{EQU}) \quad (2.12c)$$

These equations are readily implementable by using single electronic gates of the type AND, OR, XOR. We shall consider this aspect only at the end of the discussion.

For unary relations, namely of the type $\underline{a} \underline{Z} = \underline{b}$ also, such simple circuits can be designed and will be briefly described later.

The binary and unary relations are the equivalents of the following expressions in standard logic.

$$\underline{a} \underline{A} \underline{b} = \underline{c} \iff (\underline{a} \wedge \underline{b} \implies \underline{c}), \text{ for } \underline{Z} = \underline{A} \text{ etc.} \quad (2.13a)$$

$$\underline{a} \underline{I} = \underline{b} \iff (\underline{a} \wedge (\underline{a} \implies \underline{b}) \implies \underline{b}), \text{ for } \underline{Z} = \underline{I} \text{ etc.} \quad (2.13b)$$

Any set of statements in propositional calculus can be split up into a sequence of such unary and binary relations, representing the elementary steps in the implementation of the logical graph of the argument. (We will discuss the logical graph and its implementation later.)

(iii) Connection between the unary form and binary form of a relation

This can be stated very generally in the following way.

$$\text{If } \underline{a} \underline{Z} \underline{b} = \underline{c} \text{ and } \underline{c} = T, \text{ then } \underline{a} \underline{Z} = \underline{b} \text{ and } \underline{b} \underline{Z}^t = \underline{a} \quad (2.14a)$$

$$\text{If } \underline{a} \underline{Z} \underline{b} = \underline{c} \text{ and } \underline{c} = F, \text{ then } \underline{a} \underline{Z}^c = \underline{b} \text{ and } \underline{b} \underline{Z}^{ct} = \underline{a} \quad (2.14b)$$

$$\text{If } \underline{a} \underline{Z} \underline{b} = \underline{c} \text{ and } \underline{c} = D (T \oplus F), \text{ then } \underline{a} D = \underline{b} \text{ and}$$

$$\underline{b} D = \underline{a} ; |D| = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad (2.14c)$$

$$\text{If } \underline{a} \underline{Z} \underline{b} = \underline{c} \text{ and } \underline{c} = X, \text{ then either } \underline{a} \text{ or } \underline{b} = X, \text{ or } |Z| = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad (2.14d)$$

The proof of these follow from the solution of Eq.(2.10) for the pure states T and F as inputs for \underline{a} and \underline{b} . For the mixed states as inputs, e.g. $\underline{a} = T \oplus F$ we have $a_T \underline{z} \oplus a_F \underline{z}^c = \underline{b}$ which has the form $a_D \underline{z} = \underline{b}$ because matrix multiplication is linear.

Many tautologies of sentential calculus come out as identities in BA-2 matrix calculus. As indicated in Table 1, the contrapositive form of implication and the De Morgan relations follow as identities from the matrix representation.

(d) Boolean operators in SNS

These operations (\oplus and \otimes) have already been employed in matrix multiplication involving vectors and matrices. However, they have a logical consequence also when applied to two 2-vectors (\underline{a}' and \underline{a}'') forming the information (regarding the truth value of \underline{a}) coming from two different sources. The operators are :

$$\text{Union } (\underline{U}, \oplus) : \underline{a}' \underline{U} \underline{a}'' = \underline{a} \mapsto \underline{a}'_1 \oplus \underline{a}''_1 = \underline{a}_1; \underline{a}'_2 \oplus \underline{a}''_2 = \underline{a}_2 \quad (2.15a)$$

$$\text{Vidya } (\underline{V}, \otimes) : \underline{a}' \underline{V} \underline{a}'' = \underline{a} \mapsto \underline{a}'_1 \otimes \underline{a}''_1 = \underline{a}_1; \underline{a}'_2 \otimes \underline{a}''_2 = \underline{a}_2 \quad (2.15b)$$

The two operators are defined in terms of BA-1 algebra on the r.h.s of (2.15a,b). It will be noticed that they have a close resemblance to (2.12a,b) for the connectives \underline{A} and \underline{Q} . This interesting feature will be commented upon in the lecture.

.11.

Of the two Boolean connectives $\underline{\underline{U}}$ and $\underline{\underline{V}}$, the vidya operator finds great use for making consistency checks to find out whether there is a contradiction. It gives the contradictory state $X = (0 \ 0)$ if one of $\underline{\underline{a'}}$, $\underline{\underline{a''}}$ is T and the other is F. Also, it gives the pure state T or F, when one of $\underline{\underline{a'}}$, $\underline{\underline{a''}}$ is T or F, and ^{the} other is D. It is very useful in multivalued logic, to find out the common elements between two n-vectors $\underline{\underline{a'}}$ and $\underline{\underline{a''}}$.

The union operator is particularly useful for finding the logical sum of two SNS states. In the case of two SNS vectors $\underline{\underline{a'}}$ and $\underline{\underline{a''}}$, it gives T, or F, only if both of them are T or F respectively. If one of them is T and the other is F, it gives D, and also D for all other combinations, except for $\underline{\underline{X}} \underline{\underline{U}} \underline{\underline{X}} = \underline{\underline{X}}$. The 4x4 truth tables for $\underline{\underline{U}}$ and $\underline{\underline{V}}$ are given in Table 2(a,b) and it will be noticed that they cannot be represented by a matrix operator for the following reasons.

The 2x2 matrix of BA-2 uses the states T(1 0) and F(0 1) as the generators of the Boolean algebra. For the mixed states D and X, the output $\underline{\underline{c}}$ of $\underline{\underline{a}} \underline{\underline{Z}} \underline{\underline{b}}$ is expressible in terms of

the pure states T and F. Thus,

$$D \underline{\underline{A}} T = (T \underline{\underline{A}} T) \oplus (F \underline{\underline{A}} T) = T \oplus F = D \quad (2.16a)$$

$$D \underline{\underline{A}} F = (T \underline{\underline{A}} F) \oplus (F \underline{\underline{A}} F) = F \oplus F = F \quad (2.16b)$$

$$D \underline{\underline{A}} D = (D \underline{\underline{A}} T) \oplus (D \underline{\underline{A}} F) = D \oplus F = D \quad (2.16c)$$

The linear algebra of BVMF leads to these simple equations for the state D in terms of T and F. On the other hand, it will be seen that no such formula exists for the 4x4 table in Tables 2(a and b). In particular, for all matrix operators $\underline{\underline{Z}}$, if either $\underline{\underline{a}}$ or $\underline{\underline{b}}$ in $\underline{\underline{a}} \underline{\underline{Z}} \underline{\underline{b}} = \underline{\underline{c}}$ is X, then the resultant $\underline{\underline{c}}$ is also X. This still holds good for $\underline{\underline{Z}} = \underline{\underline{V}}$, but not for $\underline{\underline{Z}} = \underline{\underline{U}}$. The operator $\underline{\underline{U}}$ can lead to X for $\underline{\underline{a}} \underline{\underline{U}} \underline{\underline{b}}$ only if both $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are X.

Table 2. Truth tables in SNS for the two Boolean connectives
U and V

(a) Union					(b) Vidya				
<u>U</u>	T	F	D	X	<u>V</u>	T	F	D	X
T	T	D	D	T	T	T	X	T	X
F	D	F	D	F	F	X	F	F	X
D	D	D	D	D	D	T	F	D	X
X	T	F	D	X	X	X	X	X	X

For comparison, the corresponding 4x4 truth tables in SNS for the two matrix connectives A and O are given in Table 3.

Table 3. 4x4 truth tables for the matrix connectives \underline{A} and \underline{Q}

\underline{A}	T	F	D	X
T	T	F	D	X
F	F	F	F	X
D	D	F	D	X
X	X	X	X	X

\underline{Q}	T	F	D	X
T	T	T	T	X
F	T	F	D	X
D	T	D	D	X
X	X	X	X	X

3. Logical graphs and their implementation

(a) Elements of logical graphs

As mentioned above, any argument can be split up into a sequence of unary, binary or binary reverse equations.

These have the forms as in (3.1a,b,c)

$$\underline{a} \underline{Z} = \underline{b} , \quad \underline{a} \underline{Z} \underline{b} = \underline{c} , \quad \underline{a}(\underline{c}, \underline{Z}) = \underline{b} \quad (3.1a,b,c)$$

Each of these can be represented by an elementary logical graph as shown in Fig.1.

Therefore, any argument in SNS (propositional calculus) can be represented by ^alogical graph built up of these component building blocks. This will become particularly clear from the very simple example given below.

.14.

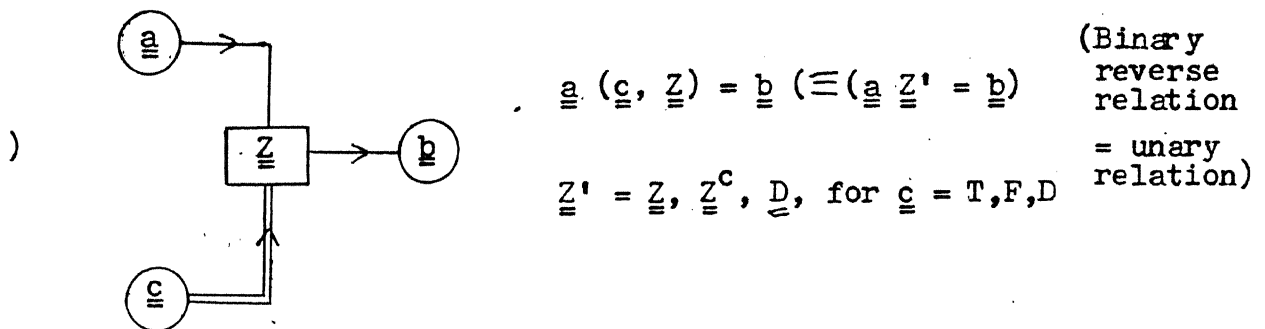
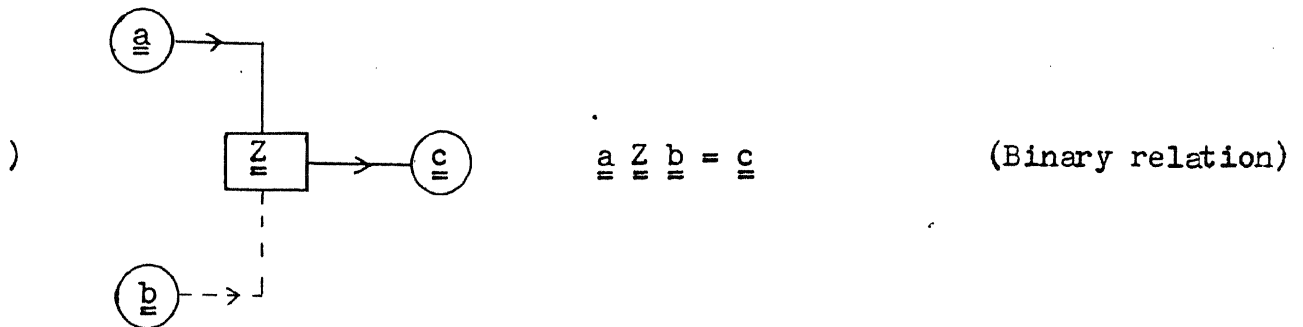
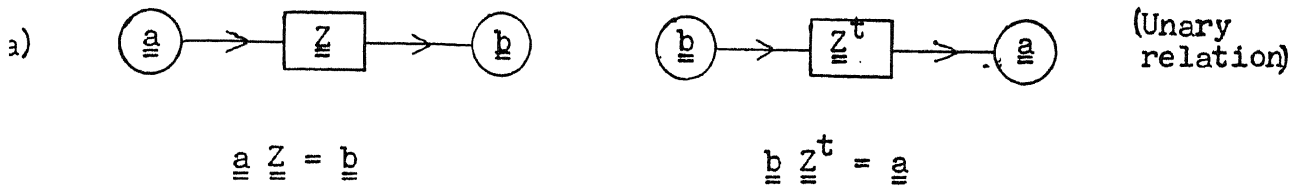


Fig.1. Building blocks of logical graphs.

(b) Simple example of a logical argument

Consider the argument "a and b implies x" i.e. $(\underline{a} \wedge \underline{b}) \Rightarrow \underline{x}$
 This can clearly be broken up into elementary statements,
 as in (3.2a,b)

$$\underline{a} \wedge \underline{b} = \underline{g} \quad (\text{binary}) \quad ; \quad \underline{g} \text{ I } = \underline{x} \quad (\text{unary}) \quad (3.2a,b)$$

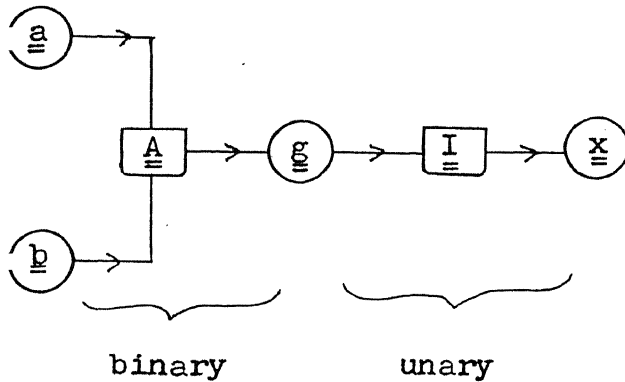
The logical graph of this is shown in Fig. 2(a), and the intermediate output g and the final output x, corresponding to inputs a and b having the truth values T and F, ^{are} given by its side. It will be seen from this, that x has a definite truth value (namely T) only for the first combination (T, T) for (a, b). The way in which this can be generalized is obvious. We shall give below in Section(d) a proof that any general argument can be split~~ted~~ up into a sequence of such elementary statements and implemented. We shall now consider some finer points in connection with implementing these graphs.

(i) Application of vidya operator : Consider the graph in Fig. 2(b). The equations are

$$\underline{a} \text{ I} \quad \quad \underline{b} \text{ I N} = \underline{x} \quad (3.3a,b)$$

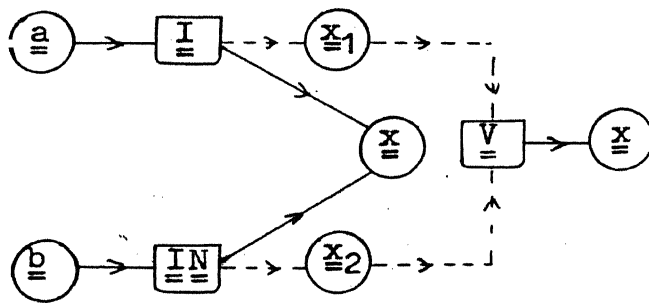
The interesting point is that there are two inputs for x coming as the outputs of the l.h.sides on (3.3a), (3.3b) respectively.

16.



\underline{a}	\underline{b}	\underline{g}	\underline{x}
T	T	T	T
T	F	F	D
F	T	F	D
F	F	F	D

(a)



\underline{a}	\underline{b}	$\underline{x_1}$	$\underline{x_2}$	\underline{x}
T	T	T	F	X
T	F	T	D	T
F	T	D	F	F
F	F	D	D	D

(b)

Fig.2. Examples of Logical graphs

(a) \underline{a} and \underline{b} are inputs, \underline{g} is the intermediate input-output, and \underline{x} is the output.

(b) Application of the vidya operator when a term \underline{x} has two inputs.

Consequently, the two may agree, or may contradict, one another. To take care of this, the vidya operator \underline{V} should be applied, as shown by dotted lines in Fig. 2(b). As mentioned in the previous section, this can not only detect the existence of contradiction, but the vidya check also gives at the same time the nature of \underline{x} which is the outcome of taking the common information in \underline{x}_1 and \underline{x}_2 . We shall illustrate this. Thus, the outputs for different combinations of T and F for \underline{a} and \underline{b} are given in the table by the side of Fig. 2(b). As may be seen from this, if $\underline{a} = T$ and $\underline{b} = T$, \underline{x}_1 and \underline{x}_2 have contradictory truth values T and F, and this is indicated by the impossible state $X = (0 \ 0)$ for \underline{x} . On the other hand, the other three combinations of \underline{a} , \underline{b} do not lead to contradiction. Taking the second row with $\underline{a} = T$ and $\underline{b} = F$, $\underline{x}_1 = T$ while \underline{x}_2 is only D, so that when \underline{x}_1 and \underline{x}_2 are combined by the vidya operator, we get the more informative of the two prevailing over the other and giving the output \underline{x} as T. This is a very general principle in logic and in proof theory, namely that if a statement cannot be proved to be either true or false by one method, and is definitely proved to be true by another, then the second result is to be taken.

(c) Reversal of argument from contradictions

The question may be asked as to how we can find out the precise origin of a contradiction and make deductions from this. We shall illustrate this by the simple example of the logical graph in Fig. 2(a). In this, suppose, we denote the output \underline{x} by \underline{x}_1 , and that we are given that the correct truth value of \underline{x} is $\underline{x}_2 = F$. Then by the construction of the vidya operator in Fig. 2(b), it is seen that $\underline{x}_1 \vee \underline{x}_2 = T \vee F = X$, indicating contradiction. The question is, what we do thereafter.

It is assumed that the latter information, namely $\underline{x}_2 = F$ is definitive. Therefore, $\underline{x}_1 = T$ is forbidden, and consequently \underline{x}_1 can be either F or D. We will take the former first and denote the new value of \underline{x}_1 by $\underline{x}' = F$. We then reverse the argument backwards and rewrite Eqs.(3.2a,b) in the reverse direction as (3.4), (3.5).

$$\underline{x}' \underline{\underline{I}}^u = \underline{g}' \quad (3.4)$$

$$\underline{a}(\underline{g}', \underline{A}) = \underline{b}' \quad (3.5)$$

We further assume that \underline{a} is given to be T without question, and the possible value for \underline{b} , denoted by \underline{b}' , is to be determined.

What is done effectively is to reverse the unary relation by using the transposed matrix, as in Fig. 1(a), and reverse the binary relation to obtain (3.5), as in Fig. 1(c). Then the steps we obtain are the following.

$$\text{In (3.4), } \underline{x}' = F \quad (0 \ 1), \quad |I^t| = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mapsto (0 \ 1) \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (0 \ 1) = \underline{g}' \\ = F \quad (3.6)$$

$$\text{In (3.5), } \underline{g}' = F \equiv (\underline{a} \ \underline{A}^C = \underline{b}') \quad \text{and} \quad \underline{a} = T \mapsto (1 \ 0) \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} = (0 \ 1) \\ = \underline{b}' = F \quad (3.7)$$

Thus, without making any logical argument, but using purely the matrix algebra of (3.4) and (3.5) which contains all the steps that are needed, we unequivocally come to the conclusion that \underline{b}' is equal to F . Since \underline{b} is an initial input and the graph cannot be traced back thereafter, we can conclude that $\underline{x} = F$ and $\underline{a} = T$ lead unquestionably to $\underline{b}' = F$.

If $\underline{x}' = D$ then the steps in (3.6) and (3.7) become as follows.

$$(3.4), \quad \underline{x}' = D \mapsto (1 \ 1) \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (1 \ 1) = \underline{g}' = D \quad (3.8)$$

$$(3.5), \quad \underline{g}' = D \equiv (\underline{a} \ \underline{D} = \underline{b}') \mapsto (1 \ 1) \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = (1 \ 1) = \underline{b}' = D \quad (3.9)$$

Thus the doubtful state for \underline{x} again leads to the doubtful state for \underline{b} , and hence is of no value, as it does not contradict the original input of $\underline{b} = T$. Therefore, if the revised input of x is F, it leads only to the conclusion that the revised truth value of \underline{b} is F.

The main point to be emphasized is that the matrix algebra that we have developed gives a very simple formalism for reversing the direction of flow of a logical argument for back-tracking the steps in a logical graph. It is very useful, for example, in finding out the precise origin of the contradiction. This is true not only in propositional calculus, but also in predicate calculus, where the graph-theoretical procedures are identical in principle, but the algebra of the logical equations may be more complicated.

(d) Proof that any argument can be implemented as a sequence of elementary relations

We shall give the proof of the theorem that the pattern of Eqs (3.2a,b) can be extended to any argument in propositional calculus (provided it does not have a circular sub-argument in it). In such a case, the process indicated in Eqs (3.2a,b) can be

applied to break up the graph into a set of unary and binary elementary relations. The result to be proved is that these relations can be rearranged so as to be sequentially implementable. We shall show that this is possible for a general graph \mathcal{G}_N representing a list containing N elementary statements, each associated with one connective \underline{Z}_j .

It is readily verified that this is possible for a small number of connectives \underline{Z}_1 and \underline{Z}_2 as shown in Fig. 3. We shall extend this to any N by induction, by showing that, if the property of sequential implementability holds for all graphs \mathcal{G}_N , then it holds for all \mathcal{G}_{N+1} , provided the graphs do not contain any directed circuits. This is possible because each equation is only either a unary or a binary relation (the topology and connectivity of the input and output to the connective \underline{Z} is the same for $\tilde{\text{binary forward}}$ and $\tilde{\text{binary reverse}}$ relation). Therefore, if Eqs. (1), (2), ... (N) are given, and Eq.(N+1) is added, the last can only be a unary or binary relation. The condition that the graph does not have any directed circuits necessarily means that there is also one output from \mathcal{G}_N which we may designate by the term x .

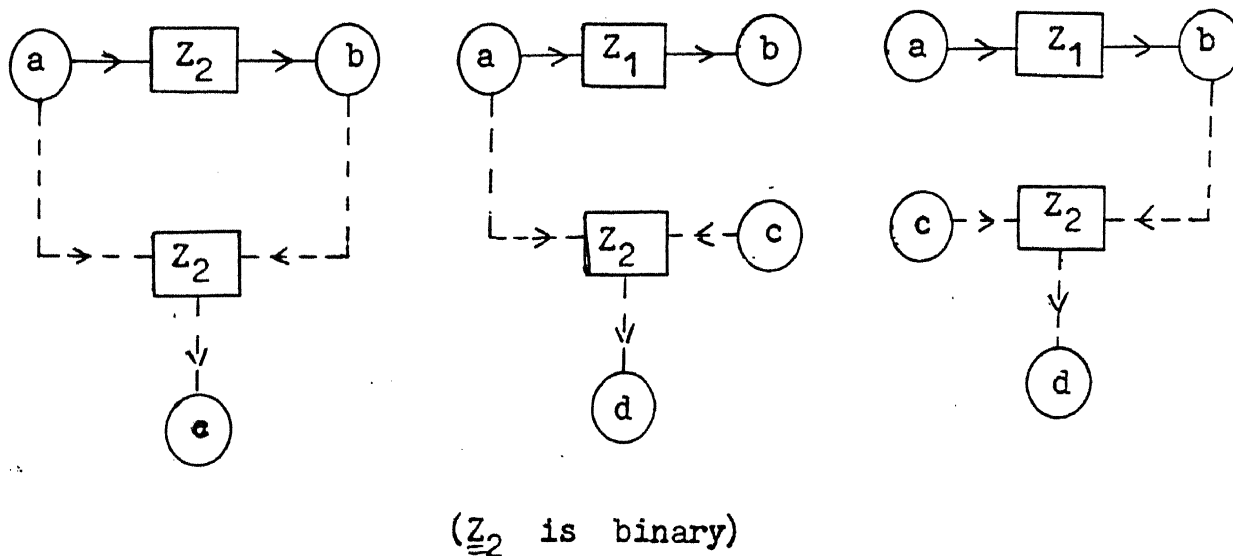
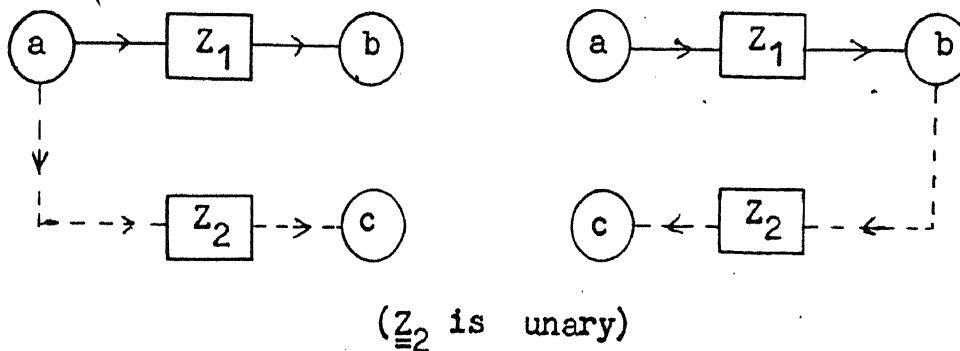


Fig.3. Possible ways of constructing G_2 from G_1 by adding \underline{Z}_2 in a unary or binary relation for Eq.(2).

Suppose Eq.(N+1) is unary, then it will have the pattern

$$\text{Eq. (N+1)} : \underline{x} \underline{Z(N+1)} = \underline{y} \quad (3.10)$$

where \underline{x} is the above mentioned-term in \mathcal{G}_N in the graph and \underline{Z} is the connective involving the (N+1)th unary relation.

Since the graph with N equations is implementable by assumption, the input \underline{x} in (3.10) is available, and therefore Eq.(N+1) is implementable.

Suppose Eq.(N+1) is a binary relation ; then it has the form

$$\text{Eq. (N+1)} : \underline{x}_1 \underline{Z(N+1)} \underline{x}_2 = \underline{y} \quad (3.11)$$

Here, one of the terms \underline{x}_1 or \underline{x}_2 , must belong to the graph \mathcal{G}_N

for the resulting graph to be connected. One such term is always available if the graph does not have directed circuits.

The other term may belong to \mathcal{G}_N , or may be a fresh input \underline{g} .

In either case, the information needed for implementing the step in Eq.(N+1), namely the truth values of the terms \underline{x}_1 and \underline{x}_2 are available from the implementation of \mathcal{G}_N which is assumed.

Therefore, the step N+1, forming the Eq.(N+1), is implementable

in sequence thereafter. Thus, we prove by induction that any graph \mathcal{G}_N representing N elementary relations can be implemented in sequential order provided it does not contain any directed circuits. In fact, this proof holds for all values of $N \geq 1$.

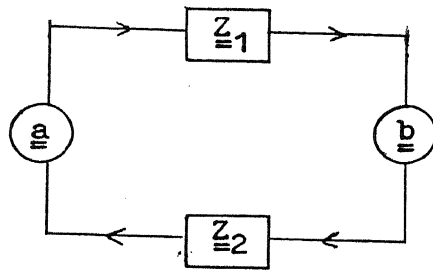
Ideas for writing an algorithm for this purpose have been developed, but not fully worked out. We shall now consider some simple examples of circular graphs which contain directed circuits and which can give rise to complications, and show that they can be modified by removing the directed circuit and substituting it by a non-directed circuit, so that they also become implementable by the above theorem. Hence, the result of this subsection is quite general for all graphs in propositional calculus. In fact, the proof can be readily extended to logical arguments in first order predicate calculus also, with the only difference that the elementary unary and binary relations have more complicated algebraic forms, but logically having the same graphical structure as in Fig.2.

(e) Implementation of circular arguments

The essential problems, involved in the construction and implementation of logical graphs containing circular arguments, become apparent even in a very simple example that is given below. We consider an argument composed of two statements as in (3.12a,b)

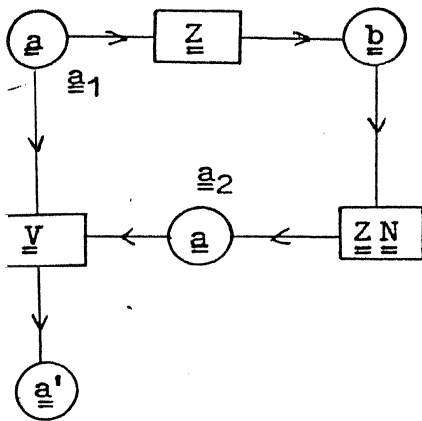
$$\underline{a} \underline{Z} = \underline{b} , \quad \underline{b} \underline{Z} \underline{N} = \underline{a} ; \quad \underline{Z} = \underline{I} \text{ or } \underline{E} \quad (3.12a,b)$$

The general graph for this is shown in Fig.4(a), and it will be seen that there is no initial input which is not also an output. Consequently, we have to choose the input to be either one of the two terms \underline{a} or \underline{b} . The two possibilities are shown in Fig. 4(b) and are labelled Case(i) and Case (ii) respectively. In Case(i), the input truth value of \underline{a} is designated as \underline{a}_1 and the output for the same term coming from the graph is labelled \underline{a}_2 . Since there are two inputs, we have to include a vidya check, and this is shown in the figure, with the net resultant value of \underline{a} marked as \underline{a}' . The graph for case (ii) is similar, with the difference that the initial input is given to \underline{b} . It is to be noted that both the graphs for case (i) and case (ii) do not have directed circuits, and their equations can be written as follows:

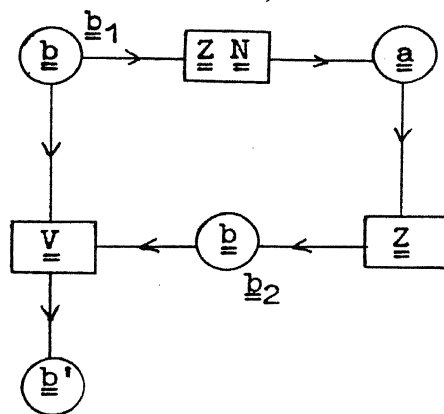


(a)

(a) $\underline{a} \underline{Z} = \underline{b}$, $\underline{b} \underline{Z} \underline{N} = \underline{a}$



Case (i)



Case (ii)

\underline{Z}	Case (i)			\underline{a}'	Case (ii)			\underline{b}'
	\underline{a}_1	\underline{b}	\underline{a}_2		\underline{b}_1	\underline{a}	\underline{b}_2	
\underline{I}	T	T	F	X	T	F	D	T
	F	D	D	F	F	D	D	F
	D	D	D	D	D	D	D	D
\underline{E}	T	T	F	X	T	F	F	X
	F	F	T	X	F	T	T	X
	D	D	D	D	D	D	D	D

(c)

Fig.4. Logical graph of a simple circular argument

(a) Equations and graph

(b) Breaking up the circuit

(c) Truth tables for the two cases in (b)

$$\underline{a}_1 \underline{Z} = \underline{b}, \quad \underline{b} \underline{Z} \underline{N} = \underline{a}_2, \quad \underline{a}_1 \underline{V} \underline{a}_2 = \underline{a}' \quad (3.13)$$

$$\underline{b}_1 \underline{Z} \underline{N} = \underline{a}, \quad \underline{a} \underline{Z} = \underline{b}_2, \quad \underline{b}_1 \underline{V} \underline{b}_2 = \underline{b}' \quad (3.14)$$

The results of applying these equations in the two cases for the connective \underline{Z} being \underline{I} (imply) and \underline{E} (equivalent) are given in the figure 4(c).

(i) Discussion for $\underline{Z} = \underline{I}$: The most interesting consequence that comes out of the table is seen from the entries for $\underline{Z} = \underline{I}$. In this case, the possible truth values of $(\underline{a}, \underline{b})$ which satisfy Eqs (3.12a,b) are different, according as the circular graph is broken at \underline{a} , or at \underline{b} . This is seen by comparing the two tables ⁱⁿ the upper half of Fig. 4(c). Case (i) only permits the combinations $(\underline{a}, \underline{b})$ to have values (F, D) and (D, D), while case (ii) permits the combinations (F, T), (D, F) and (D, D). In fact, apart from the completely non-informative combination (D, D), the output indicating the permissible truth values of \underline{a} and \underline{b} is quite different for case (i) and case (ii), although they both correspond to the same logical graph of Fig. 4(a). Therefore, one important conclusion made is that more specifications should be given whenever there is a directed

circuit in a logical graph. In particular, the point at which the directed circuit is to be broken should be specified as one of the input information for the implementation of the argument. If there are more than one directed circuits, then every one of them must be broken, and a vidya check introduced at ^{each} point of breakage, comparing the input and output of the term at which the circuit is broken. No doubt this will lead to a non-directed circuit, but then the graph is implementable by the procedures mentioned earlier.

(ii) Discussion for $Z = E$: The lower half of the Table 3(c) is still more interesting, in that, irrespective of whether the circular graph is broken at \underline{a} , or at \underline{b} , no combination of definite truth values T or F is possible for $(\underline{a}, \underline{b})$. Thus, in case (i), both $\underline{a}_1 = T$ and $\underline{a}_1 = F$ lead to the contradictory state X for \underline{a}' , and similarly, in case (ii), both $\underline{b}_1 = T$ and $\underline{b}_1 = F$ lead to the contradictory state X. In fact, this logical graph is ^aslightly modified version of the classical paradox of the Barber of Seville. We may call this as the ^{statement} "double/ paradox". In this case, reversing the argument from

the contradiction does not lead to the removal of the contradiction, because both $\underline{a} = T$ and $\underline{a} = F$ lead to contradictions and similarly for $\underline{b} = T$ and F . (in case (i)), / However, it should be pointed out that this does not mean that there is no solution to the problem, because the input $\underline{a}_1 = D$ does not lead to a contradiction, and both \underline{a} and \underline{b} can have the truth value D . Thus, although the doubtful state D represents the state of tautology in standard logic, it does have information value in SNS algebra.

Therefore, the statement "may be true or may be false, is the maximum that we can say about \underline{a} and \underline{b} " , is a factual statement/

In fact, this should be augmented by saying that the truth value of both \underline{a} and \underline{b} is " D , but not T and not F " . This state does exists in BA-3 algebra as we will show later, and is quite a reasonable Boolean algebraic notion which is essentiall for our BA-3 representation of quantifier states.

CONTENTS

Page No.

Lecture 2 : Logical graphs and their implementation

1. Introduction	1
2. Implementation of logical arguments in SNS	2
(a) Rules for the forward implementation	3
(b) Location of the consistency checks <u>via</u> the vidya operator and for conjunctions and their immediate consequence	5
(i) Vidya operator	5
(ii) Unary relations	6
(iii) Binary reverse relation	8
(iv) Summary of Section 2(b)	10
(c) Procedure for the back-tracking from a contradiction to its ultimate source	11
(i) Unary relation	12
(ii) Binary reverse relation	13
(iii) Binary forward relation	14
3. Example of a simple argument in SNS	15
(a) Rearrangement of the list of equations for sequential implementation	15
(b) Implementation of the sequential listing in the forward direction	19

Logical graphs and their implementation

1. Introduction

This lecture will deal with the principles involved in the construction of the logical graph corresponding to a given argument and rearranging the steps of the argument so as to be sequentially implementable. This is discussed in Section 2 and for this purpose we need the principles developed in Section 3 of Lecture 1 — in particular the idea that any argument can be split up into a sequence of unary, binary or binary reverse equations, of the form of Eqs.(3.1a,b,c) of Lecture 1, and that they can be represented graphically as in Fig.1 of that section.

As mentioned therein, it is possible to rearrange a given sequence of elementary relations, forming the contents of the split up argument, so that they are sequentially implementable. Therefore, the implementation of the argument can be carried out as per the rules given in Section 2(a). below, which gives in summary the procedures to be adopted, for implementing an argument so as to obtain the final outputs from the given

initial inputs (provided there are no contradictions), and for checking for the existence of any contradictions in between, or in the final output. Thereafter, in case there is any contradiction, it is necessary to retrace the graph to find out the source of the contradiction. The procedure for this can also be given, using the rearranged list, and retracing the graph in stages, in the same way as it was implemented in the forward direction in stages. The details of this ^{back-tracking} procedure are described in Section 3(c).

After giving an example of a graph, with 14 connectives and 7 stages, both for the forward implementation and the backward tracing from a contradiction, the principles governing these are discussed in Section 3. It is believed that this procedure can be applied for implementing any argument in predicate calculus, as well as in propositional calculus, since the principle of graph construction and implementation, in terms of unary, binary and binary reverse relations, is the same whether it is applied to propositional calculus, predicate calculus (or any type of mathematical or logical argument). Therefore, these principles appear to be

of great value for application to the construction of programs in computer science in general. It might even have application in the design of compilers in which it might be possible to rearrange the steps in case a syntactical error is noticed. These are matters that have to be taken up for study in due course.

2. Implementation of logical arguments in SNS

(a) Rules for the forward implementation

In the light of what was discussed in the previous lecture, and Section 1, we can formulate the following procedures for implementing an argument.

(i) Split up the argument into elementary relations of the types unary, binary forward, and binary reverse, and list them in arbitrary order.

(ii) Construct the logical graph, either in diagram or in a listing. Check for directed circuits, and if so decide where each circuit is to be broken, and introduce a vidya check between the two terms produced at each point of breakage.

(iii) Similarly introduce vidya checks wherever there are two inputs into a term. Add these steps in (ii) and (iii) to the list of elementary relations to be implemented. We shall call the new set as the enlarged set of logical equations.

(iv) Rearrange the enlarged set for sequential implementation (algorithm for this will be outlined in Section 3). At every place, where the output of a relation comes from the operator \underline{V} or of $\underline{A}(k, \ell)$ as a unary, or a binary reverse operator, add a statement to check if the output is $X = (0 \ 0)$. (The reason for this is explained below in subsection 3(a).)

(v) Implement the logical equations one by one in sequence, including the X-checks. If there is no X as the output of any equation, proceed to the next equation in succession, until the list is exhausted.

(vi) If, at any step, the output is X, then stop further calculations and check for the source of the contradiction by retracing the graph. The details of the procedure for this are described in Section 3, after the illustration of the above in section 2(b), procedure is given in an example, and the retracing is explained in an intuitive manner.

(b) Location of the consistency checks via the vidya operator and for conjunctions and their immediate consequence

We shall explain the reason why the X-checks need be included only at certain places of the implementation in the above rules of procedure. This is because of the property that the matrix connectives of the $\underline{O(I)}$ -type, and of the \underline{E} -type, can never yield the impossible state X as the output of a unary or binary relation, unless X is also an input, which is forbidden as mentioned above, since the calculation stops when X is the output of any equation. It is also true for $\underline{A(k, \ell)}$ employed in a binary forward relation. So also, the Boolean connective \underline{U} does not lead to X, as output, except when both the inputs are X, which is again forbidden. Therefore, the impossible state X can be the output of an equation in the sequence of operations only if it is concerned with a binary reverse, or a unary relation, involving a conjunction $\underline{A(k, \ell)}$, or the vidya operator \underline{V} employed for the consistency check.

(1) Vidya operator

The case of the vidya operator has already been discussed in Section 3(b) of Lecture 1, and it has been shown there that it will give X as output for non-X inputs only if one input

is T and the other is F. (This is the standard check that is used for detecting contradictions, but we definitely look for it by obtaining it as the output of the Boolean connective $\underline{\vee}$.)

(ii) Unary relations

As mentioned above, contradictions can occur for unary relations only in the case of $\underline{A}(k, \ell)$ as the operator since both $\underline{O}(\underline{I})$ -type and \underline{E} -type connectives can never lead to (0 0) as output. For the case of $\underline{A}(k, \ell)$, we may illustrate the occurrence of X by the example of the unary relation $\underline{a} \underline{A} = \underline{b}$. In this $\underline{A} \equiv \underline{A}(1, 1)$, so that, if $s(ka) = F = (0 \ 1)$, we obtain

$$\underline{a} \underline{A} = (0 \ 1) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} (0 \ 0) = X = \underline{b} \quad (1)$$

However, the contradiction which is indicated by X coming out as the output of Eq.(1), is really due to the contradiction between the input $\underline{a} = F$ and the conjunction relation $\underline{a} \wedge \underline{b}$, which is what is indicated by the equation $\underline{a} \underline{A} = \underline{b}$ (or $\underline{a} \underline{A} \underline{b} \quad T$). The latter requires that both \underline{a} and \underline{b} must be T, and this contradicts the input $\underline{a} = F$. Consequently, the retracing process will begin by making a revision of $\underline{a} = F$ to $\underline{a}^r = T$

and tracing the graph backwards to the origin of the contradiction (either in one, or more, of the inputs, or in some other way, as will be seen below). Obviously, the revised value of \underline{a} , namely T, does not lead to contradiction in Eq.(1), but that is not important in so far as retracing is concerned.

This simple case of Eq.(1) can be generalized to the form of Eq.(2) (See p. A1)

$$s(ka), \underline{a} \underline{A}(k_1, \ell_1) = \underline{b} \mapsto s(\ell b) = (0 \ 0) = X, \text{ if } k = k_1^C \quad (2)$$

Here also, the detection of the contradiction is made by the occurrence of the output X of the particular equation in (2); but its real origin is in the contradiction between the input $s(ka)$ and the relation $s(k_1) \underline{a} \underline{A}(k_1, \ell_1) s(\ell_1) \underline{b}$.

This feature of a possible contradiction between the input and the unary relation of the conjunctive type to which it is applied, occurs not only in propositional calculus, but, as we shall see later, also for relations involving quantifiers, of the QL-1 type and QL-2 type, in predicate logic, and is a common feature that is observed in the implementation of any logical graph.

(iii) Binary reverse relation

We have already seen that a binary relation $\underline{a}(\underline{c}, \underline{z}) = \underline{b}$ is equivalent to a unary relation $\underline{a} \underline{z}' = \underline{b}$, with $\underline{z}' = \underline{z}$ or \underline{z}^c , according as $\underline{c} = T$ or F . We have also seen that the equation $\underline{a} \underline{z}' = \underline{b}$ can lead to a contradiction only if \underline{z}' is of the type $\underline{A}(k, \ell)$. Hence, for binary reverse relations, contradictions can arise only for two types of equations illustrated by (3a) and (3b).

$$\underline{a}(\underline{c}, \underline{A}) = \underline{b}, \underline{c} = T, \text{ with input } \underline{a} = F \mapsto \underline{b} = X \quad (3a)$$

$$\underline{a}(\underline{c}, \underline{O}) = \underline{b}, \underline{c} = F, \text{ with input } \underline{a} = T \mapsto \underline{b} = X \quad (3b)$$

In both cases, the contradiction arises in the same manner as in Eq.(2), with the effective unary relation that is operative being $\underline{a} \underline{A}(k_1, \ell_1) = \underline{b}$, and the input $s(ka)$ being such that $k = k_1^c$. We only give the theory here, but for practical purposes, the BVM formulae for Eqs (3a) and (3b), for the general case of $\underline{A}(k, \ell)$ and $\underline{O}(k, \ell)$ will automatically take care of all these and indicate X when there is contradiction. It will also automatically mean that the contradiction has arisen because the input to a conjunctive unary relation is not consistent

with the states of \underline{a} and \underline{b} that are demanded by the conjunction. Therefore, all that is necessary for removing the contradiction is to convert \underline{a} to the revised value $\underline{a}^r = \neg \underline{a} = s(k^c a^r)$, and retrace the graph with the revised value of \underline{a}^r .

The main difference between the unary relation considered in (ii) and the binary reverse considered in this subsection (iii) is that there is a second alternative by which the contradiction can be removed, namely modifying the input \underline{c} into $\underline{c}^r = \neg \underline{c}$. The consequence of this can be illustrated by the examples in (3a) and (3b), as will be seen in (4a) and (4b). Thus, taking (3a) with $\underline{c}^r = F$, we obtain the equivalent unary relation and its consequence in (4a)

$$\underline{a} \underline{0}(2,2) = \underline{b}, \text{ with } \underline{a} = F \mapsto \underline{b} = F (\neq X) \quad (4a)$$

Similarly, taking the binary reverse relation in (3b), and putting $\underline{c}^r = T (= \neg \underline{c})$, we obtain

$$\underline{a} \underline{0}(1,1) = \underline{b}, \text{ with } \underline{a} = T \mapsto \underline{b} = T (\neq X) \quad (4b)$$

Eqs (4a) and (4b) indicate that the change from \underline{c} to $\underline{c}^r = \neg \underline{c}$ eliminates the contradiction in the forward direction, and it is only necessary to trace ^{at} the graph from the modified term \underline{c}^r backwards to arrive/ to the ultimate origin of the contradiction.

.10.

(.v) Summary of Section 2(b).

Thus, we have shown that the impossible state X can only arise under two situations — (a) as the output of a vidya check, and (b) as the output of a unary relation involving a conjunction. In either case, the origin of the contradiction is at input stage of the step (equation) for which X is the output. The procedure to be adopted for back-tracking in each of these cases for the minimum next step are as follows.

If $\underline{x}, \underline{v} \underline{x}, = x = X$, then \underline{x}_1 and \underline{x}_2 are mutually contradictory, e.g. $\underline{x}_1 = T, \underline{x}_2 = F$.

In that case, we must decide from the conditions of the problem as to which of the two is correct (suppose $\underline{x}_2 = F$ is correct). Then, the other one is changed to $\underline{x}_1^r = \neg \underline{x}_1$ (in this case $\underline{x}_1 \mapsto \underline{x}_1^r = F$) and the graph retraced therefrom upto the input stage.

If $\underline{a} \underline{A}(k, \ell) = \underline{b} = X$, then, by the discussion given above, it follows that \underline{a} must be changed from $s(ka)$ to $s(k^c a)$ i.e. if $\underline{a} = T$, then the revised value is $\underline{a}^r = F$, and, vice-versa, if $\underline{a} = F$, then $\underline{a}^r = T$. In this case also, we retrace the graph from \underline{a}^r in the reverse direction to the ultimate source of the contradiction.

If $\underline{a}(\underline{c}, \underline{A}(k, \ell)) = \underline{b}$ and $\underline{c} = T$, or $\underline{a}(\underline{c}, \underline{O}(k^c, \ell^c)) = \underline{b}$ and $\underline{c} = F$, we obtain effectively $\underline{a} \underline{A}(k, \ell) = \underline{b}$, and hence it is possible to have $\underline{b} = X$, as indicated in the last para. Then, we can remove the contradiction by changing either $s(k)$ of \underline{a} to $s(k^c)$ of \underline{a}^r and retrace the graph as for the unary relation, or change $\underline{c}^r = \neg \underline{c}$ and retrace the graph from \underline{c}^r to the source of the contradiction.

In all cases, the graph is to be retraced backwards, in the reverse direction to the forward implementation, from a term in the stage previous to the stage at which the contradiction arises. The actual process for doing this is described in the next subsection and all these procedures will be illustrated in a fairly complicated graph in Section 3.

(c) Procedure for the back-tracking from a contradiction to its ultimate source.

We have seen, in the above subsection (b), how we obtain the effective reversal of sign of one or more terms as a result of contradiction arising from the operators \underline{V} and $\underline{A}(k, \ell)$. We shall now consider the effect of this back-tracking process on the three types of relations that are possible, namely unary, binary forward, and binary reverse.

(1) Unary relation

Suppose, we have $\underline{a} \underline{Z}(k, \ell) = \underline{b}$ and \underline{b} is replaced by $\underline{b}^r = \neg \underline{b}$; we wish to find the effect on \underline{a} by reversing this equation. This is worked out by reversing the unary relation and writing it in the form $\underline{b}^r \underline{Z}^t(k, \ell) = \underline{a}^r$, by the simple rule that the operator for the reverse unary relation is the transpose of the forward matrix operator. We do not have to consider the transpose operator quite generally, but restrict ourselves to the case when $\underline{b}^r = \neg \underline{b}$. Then the following consequences occur as indicated in (5) and (6).

$$\text{For } \underline{Z} = \underline{O}(\underline{I})\text{-type and } \underline{E}\text{-type, } \underline{b}^r \underline{Z}^t(k, \ell) = \underline{a}^r = \neg \underline{a} \quad (5a)$$

This follows from the well-known equivalences

$$(\underline{a} \Rightarrow \underline{b}) \equiv (\neg \underline{b} \Rightarrow \neg \underline{a}) \quad \text{and} \quad (\underline{a} \equiv \underline{b}) \equiv (\neg \underline{b} \equiv \neg \underline{a}) \quad (5b,c)$$

and the fact that \underline{b}^r could not have been D, if the reversed path from a contradiction has reached it. Consequently, it is only necessary, in the retracing process, to negate the input \underline{a} to $\underline{a}^r = \neg \underline{a}$ if the output \underline{b} is negated, and the preceeding step can then be implemented for the back-tracking process.

However, the situation is quite different for $\underline{Z} = \underline{A}$ -type, for then

$$\underline{a} \underline{A}(k, \ell) = \underline{b} \neq X \mapsto \underline{b}^r \underline{A}^t(k, \ell) = \underline{a}^r = X, \text{ for } \underline{b}^r = \neg \underline{b} \quad (6)$$

This is because, in the forward direction, \underline{b} was the output of the relation $\underline{A}(k, \ell)$ and hence has the sign $s(\ell \ b)$ which is consistent ^{with} the conjunction $\underline{A}(k, \ell)$, ^{so that} when we replace it by $\underline{b}^r = s(\ell^c \ b^r)$ it obviously contradicts the relation of the reversed relation and indicates this by the output \underline{a}^r /becoming equal to X. So here we obtain an impasse. We have started retracing from a contradiction, and on going back, we once again come to a contradiction; and retracing it once again to go in the forward direction would lead back to the first contradiction. In other words, the argument itself is invalid, and some step in the argument is wrong in between first ^{contradiction} / from which the back-tracking arises and the second contradiction which is noted via Eq.(6). Such features will be discussed quite generally in connection with the discussion of paradoxes in a later lecture.

(ii) Binary reverse relation

We consider the binary reverse relation, before the forward relation, because it is effectively equivalent to a unary relation as pointed out above. Therefore, ^{if ~} / binary reverse relation leads to $\underline{a} \ \underline{Z}_1 = \underline{b}$, where $\underline{Z}_1 = \underline{Z}(k, \ell)$ as in

subsection (i) for unary relation given above, all the discussion under that heading is operative, so long as the truth value of \underline{c} is not modified. Therefore, one consequence of changing \underline{b} to $\underline{b}^r = \neg \underline{b}$ is that \underline{a} becomes $\underline{a}^r = \neg \underline{a}$, provided \underline{c} is unchanged. Here again, $b \neq D$, so that $\underline{b}^r = \neg \underline{b} \neq \underline{b}$.

On the other hand, \underline{c} may be changed to $\underline{c}^{r*} = \neg \underline{c}$, when it will be found (proof left to reader) that the forward relation gives \underline{b}^r consistent with $\neg \underline{b}$, and we can retrace from \underline{c}^{r*} to the source of the contradiction. In other words, in the case of a binary reverse relation, there are two ways of eliminating the contradiction, namely by changing either \underline{b} to $\neg \underline{b}$, or \underline{c} to $\neg \underline{c}$. Each of these possibilities will have to be worked out to the next previous stage, and successively back to the origin of the contradiction. This is quite reasonable, since the relation is a binary relation with two inputs \underline{a} and \underline{c} leading to the output \underline{b} , which is negated in the back-tracking process. We shall illustrate this in the example.

(iii) Binary forward relation

In this case, $\underline{a} \underline{Z} \underline{b} = \underline{c}$ is the equation in the forward direction from stage j to stage $j+1$ and the output \underline{c} in stage $j+1$ is converted into $\underline{c}' = \neg \underline{c}$ in the back-tracking process. Consequently, the relation becomes modified as shown in (7).

$$(\underline{a}' \underline{Z} \underline{b}' = \underline{c}' = \neg \underline{c}) \equiv (\underline{a}'(\underline{c}', \underline{Z}) = \underline{b}') \text{ and } (\underline{b}'(\underline{c}', \underline{Z}^t) = \underline{a}') \quad (7a, b)$$

3. Example of a simple argument in SNS(a) Rearrangement of the list of equations for sequential implementation

The logical^{graph} of the argument which we shall employ for the illustration is shown in ^{Sheet 11} and the listing of the elementary equations which constitute this argument is given in an arbitrary order in column 2 of Table 1^{in Sheet 12.} The graph has no directed circuits, and therefore step (i) of the rules of procedure for implementing forward arguments given in the previous section is not needed. This does not, however, make the example specialized, since there exist a number of non-directed circuits, and it has also two places where the vidya operator is employed. Thus, Eqs (7 and 8) have initially the same output d3, and Eqs (9 and 10) have the same output e1 respectively, as shown by dotted lines in Fig. 4. By step (iii) in the rules, vidya checks are introduced in both these cases, and the two inputs, of (7 and 8) and (9 and 10), respectively, are denoted by d3', d3'' and e1', e1''. Two vidya checks are then introduced, in Eqs (8a) and (10a), leading to the required outputs, d3 and e1, respectively. Therefore, the enlarged list contains 14 equations — 1 to 12, with (8a) and (10a) being added.

The procedure for step (iv) in the rules — namely the process of rearranging the enlarged set for sequential implementation — is described in a tabular manner in Table 1, and is carried out in seven stages for this particular graph, such that,

at each stage, the input(s) for the equation implemented at that stage consists of terms^s belonging to either the initial inputs given at the top of the Table, or the outputs of the previous stages, only. This can be carried out in a systematic manner, following the procedure we shall mention below in an intuitive manner. The detailed algorithm to carry this out is reserved for a later report.

(i) Stage 1 : Thus, in stage 1, we consider all equations in which either the only input for the equation is available and either one, or both, ^{of} the inputs ^{the} among the inputs a₁ to a_t are available, / This happens, for instance, in Eqs (1), (2), (3), (4), (5), (7), (8), (11) in stage 1. These are noted in ^{column under} the / number 1 of stage 1. Of these, however, only Eqs. (1), (2), (5), (8) are implementable, because the full information required for the l.h.s of the equation is available from the initial inputs, ^{only for these inputs.} The other four, namely (3), (4), (7), (11), are all binary relations in which only one of the two inputs is available from the initial data. Therefore, we mark them as "Hold". For the remaining equations, namely (8a), (9), (10), (10a) and (12), both the inputs are not available, and they are marked by a dash.

Stage 2 : In stage 2, we consider the equations that have not been implemented, and examine first those that are listed as "hold" and verify if the second term required is available in the outputs of the equations considered in stage 1. This happens, in this example, for Eqs. (3) and (4) whose outputs c2 and d1 are marked. On the other hand, it does not happen for Eqs (7) and (11) which are marked "hold" in stage 1, and therefore, once again, ^{they are} marked as "hold" in stage 2.

Then, we examine if the only input required, or one of the two inputs required, are available within the outputs of the previous stage^{1,} in any of the remaining equations. This search brings out Eq.(6), in which b2 is available from stage 1, but not d1 . Therefore, it is ^{also} marked as "hold" for stage 2. For Eqs. (8a), (9), (10), (10a) and (12), both the inputs are still not available, and they are marked by a dash. This completes stage 2.

Stage 3: The procedure adopted for stage 2 is repeated for stage 3, and we examine those marked "hold" for stage 2 first, and verify whether the second input is available in stage 2. This happens, for instance, for Eq.(6), for which d1 is

available in stage 2, and is implemented in stage 3. On the other hand, for Eq.(7) and (11), the second input is still not available and they continue to be marked as "hold".

Considering fresh inputs from stage 2, we have Eq.(9) for which c2 is available as input from stage 2, and is a unary relation, so that it is implementable in stage 3. The others are such that neither of the two inputs are available.

Stage 4: In this stage, we repeat the same process as in stages 2 and 3, and we find that Eqs (7) and (10) are implementable, while (10a) enters as "hold" and (11) continues in the category "hold".

Stage 5: In this stage, Eqs (8a) and (10a) become implementable, while Eq.(11) still continues to be "hold", and it is implemented only in the next stage.

Stage 6: Only one equation is implemented, namely Eq.(11), while (12) comes under the category "hold", and this equation is implemented in the last stage 7.

Stage 7: Eq.(12) is implemented and completes the list.

As will be readily seen, the procedure is very simple and algorithmic, and requires, at each step, only the examination of those equations not included in the sequential list, and the outputs of just the previous stage, in addition to those in the equations that are under "hold". Another merit of this process of listing them under different stages is the fact that in implementation, as will be seen below, all the equations in one stage can be implemented in parallel processing, and only the stages have to be successively implemented in serial processing. The sequence of the equations contained in a particular stage is of no importance, and can be modified, if necessary.

(b) Implementation of the sequential listing in the forward direction.

(Sheet 13)

Table 2/gives the sequential listing prepared for using the data in Table 1. implementation,/ It lists, in sequence, the inputs, and the set of equations to be implemented in each stage. Those, having the same stage number n , are given the serial numbers $(n.1)$ to $(n.k_n)$ where k_n is the number of equations considered in stage n . Then, the implementation is carried out in the sequence 1.1 to $1.k_1$, 2.1 to $2.k_2$,, and it will be found that they are sequentially implementable, as we shall illustrate in this example.

The purpose of this section is to illustrate the procedure of sequential implementation in the forward direction, and also to indicate how one can retrace the graph from a contradiction and obtain a revised set of inputs for which the contradiction will be eliminated. The former is shown in the first row of Table 3(a), in Sheet 14, in which the implementation goes through the stages, 1, 2 and 3, but leads to a contradiction in stage 4, after which the implementation is stopped. Then, we can trace back from the contradiction and this is indicated in rows 1 to 4 in the second half of Table 3(b). The resultant modified inputs a1 to a5 are then fed back in rows 2 and 3 in Table 3(a), when it is found that the graph is fully implementable, and the output x is a non-contradictory state. (The elimination of contradiction only assures that the graph will be implementable upto stage 4, where the contradiction was noticed, but luckily it is also found to be implementable upto stage 7 in this example.)

We shall only give the segments of the forward tricky/ procedure in this section; we shall process but/describe in detail the back-tracking/in the next subsection (c), since that is the one in which BVMF procedures, and the technique of sequential implementation via stages which is outlined here, are particularly interesting.

we

Thus/shall not give the vector-matrix formulae for

the implementation of Eqs. (1.1) to (1.4), (2.1), (2.2);
(3.1), (3.2) leading to the output x for $\underline{d3'}$ in (4.1),
except for the last equation (4.1) in stage 4, namely

$$\underline{a3} (\underline{d2}, \underline{0}) = \underline{d3'} \quad (8)$$

In this, $\underline{d2} = F$, so that Eq.(8) becomes effectively the unary
relation

$$\underline{a3} \underline{A}(2,2) = \underline{d3'} \quad (9)$$

In Eq.(9), $\underline{a3} = T$, as given by the listing of the input in
column 2 for inputs. This input T is in contradiction with
conjunction denoted
the 1/15 by $\underline{A}(2,2)$, which stands for the relation $\neg \underline{a3} \wedge \neg \underline{d3'}$.
This conjunction is violated by the input $\underline{a3} = T$. (This is
a very interesting feature of unary relations which was
described and explained in Section 2(c).) Hence, if $\underline{d3'}$ is
not to be X, $\underline{a3}$ must be F, and we have come back straight to
one of the inputs. Thus, one set of inputs that will remove
the occurrence of the contradictory state for $\underline{d3'}$ is

$$\underline{a1} = T, \underline{a2} = T, \underline{a3} = F, \underline{a4} = T, \underline{a5} = T \quad (10)$$

This is given in row 2 of Table 3 with the changed term $\underline{a3} = F$

marked in ring. The forward implementation of the graph with these inputs is also indicated by the truth values for the various terms occurring as intermediate input-output in the implementation of the graph. It will be noticed that $\underline{d3}'$ becomes D, so that there is no contradiction; but what is more interesting is the fact that, on going further into the implementation of stages 5, 6, and 7, there are ^{no} more contradictions occurring and the final output \underline{x} has the truth value T. It is interesting to note that although in stage 4 the two outputs $\underline{d3}'$ and $\underline{e1}''$ are both have the doubtful state D, they get purified in the successive stages to one of the pure (T or F) states/ and the output has a definite truth value^T and not the indefinite truth value D.

On the other hand, we can also have $\underline{d2}$ changed from F to $\underline{d2}^r = T$ when Eq.(8) leads to Eq.(11) below.

$$\underline{a3} \underline{0}(1,1) = \underline{d3}^r, \quad \underline{a3} = T \mapsto \underline{d3}^r = T \neq X \quad (11)$$

thus removing the contradiction at $\underline{d3}$. We should therefore

retrace the graph from the revised value of $\underline{d2}^r = T$. The

of Table 2

relevant equation is (3.1), namely $\underline{d1} \underline{A} \underline{b2} = \underline{d2}$. In this;

as a result of reversal of the graph, $\underline{\underline{d2}}^r$ has been found to have the value T. Therefore, in order to reverse the graph from here backwards, the necessary condition is

$$\underline{\underline{d1}}^r \wedge \underline{\underline{b2}}^r = T \mapsto \underline{\underline{d1}}^r = T \text{ and } \underline{\underline{b2}}^r = T \quad (12)$$

(Here we are using an intuitive argument rather than the strict BVMF procedure as given in Fig. 3(c) and Section 2(c)(iii). However, the idea is clear that in this path of the reversed graph, two changes have to be simultaneously executed, namely of changing $\underline{\underline{d1}}$ from F to T and changing $\underline{\underline{b2}}$ from F to T. We shall follow the consequences of each of these.)

Thus, taking $\underline{\underline{b2}}^r = T$, we note that $\underline{\underline{b2}}$ is the output of the equation (1.3) $\underline{\underline{a3}} \text{ I}(1,2) = \underline{\underline{b2}}$ in stage 1. Therefore, on reversing it with $\underline{\underline{b2}}^r = T$, we have

$$\underline{\underline{b2}}^r \text{ I}(1,2) = \underline{\underline{a3}}^r, \quad \underline{\underline{b2}}^r = T \mapsto \underline{\underline{a3}}^r = F \text{ (Input term)} \quad (13a)$$

On taking $\underline{\underline{d1}}^r = T$, we must reverse Eq.(2.2) of stage 2, namely

$\underline{\underline{c1}} \text{ O}(2,2) \underline{\underline{a3}} = \underline{\underline{d1}}$, in which the value $\underline{\underline{d1}} = F$ in the forward implementation has to be changed to $\underline{\underline{d1}}^r = T$. Hence we obtain the reversed equation

$$(\underline{\underline{c1}}^r \text{ O}(2,2) \underline{\underline{a3}}^r = T) \equiv (\underline{\underline{c1}}^r \wedge (1,1) \underline{\underline{a3}}^r = F) \mapsto \underline{\underline{c1}}^r = F \text{ or } \underline{\underline{a3}}^r = F \quad (13b)$$

Of these, $\underline{a3}^r = F$ is a revised input and is the same as what was obtained in (13a).

Therefore, taking $\underline{c1}^r = F$ we reverse the equation (1.2) namely $\underline{a1} \underline{0} \underline{a2} = \underline{c1}$, which leads to (13c).

$$(\underline{a1}^r \underline{0} \underline{a2}^r = F) \equiv (\underline{a1}^r \underline{A}(2,2) \underline{a2}^r = T) \Rightarrow \begin{matrix} \underline{a1}^r = F & \text{and} & \underline{a2}^r = F \\ \text{(inputs)} & & \text{(13c)} \end{matrix}$$

Thus, the second possibility of having $\underline{b2}^r = T$ leads to the consequences for the input of (13a), (13b) and (13c) with these conditions interconnected by the logical connectives "OR" and "AND". These results are indicated schematically in Fig. 5 ^{in sheet 15,} / and also in a tabular manner in Table 3(b) from which it can be seen that they lead only ~~to~~ either ~~the~~ the set of revised inputs already considered in row 2 ^{of Table 3} / or the set of inputs listed in row 3, namely $\underline{a1} = F$, $\underline{a2} = F$, $\underline{a3} = F$, $\underline{a4} = T$, $\underline{a5} = T$.

Just as with the inputs in row 2, in the case of inputs in row 3 also, the graph is fully implementable. Thus, the contradiction at $\underline{d3}'$ is removed and replaced by truth value D. However, the final output is also in the doubtful state D. The important point is that the contradiction has been eliminated and the graph shown to be implementable upto stage 4.

It is not difficult to convince ourselves that the inputs in row 2 and row 3 are the only ones with $\underline{a_4} = T$ and $\underline{a_5} = T$ that will avoid the contradiction X. To illustrate this in row 4, the graph is implemented for the inputs in (15) $\underline{a_1} = F$, $\underline{a_2} = T$, $\underline{a_3} = F$, $\underline{a_4} = T$, $\underline{a_5} = T$, and it will be seen that X arises again at d_3 .

$$s(1) = T = (1 \ 0) \ ; \ s(2) = F = (0 \ 1)$$

$$s(3) = D = (1 \ 1) \ ; \ s(4) = X = (0 \ 0)$$

$$s(ka) \equiv (\underline{a} = s(k)) \ ; \ s(ka) \underline{Z}(1,1) \ s(\ell b) \equiv \underline{a} \underline{Z}(k, \ell) \underline{b}$$

$$s(ka) \underline{Z}(k_1, \ell_1) = s(\ell b) \quad (\text{unary})$$

$$s(ka) \ s(\ell b) \underline{Z}(k_1, \ell_1) = s(mc) \quad (\text{binary forward})$$

$$s(ka), (s(m_1c), \underline{Z}(k_1, \ell_1)) = s(\ell b) \quad (\text{binary reverse})$$

$$\equiv s(ka) \underline{Z}'(k_1', \ell_1') = s(\ell b)$$

$$s(1a) \underline{A}(1,1) = s(1b) \ ; \ (1 \ 0) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad (1 \ 0) = T$$

$$s(2a) \underline{A}(1,1) = s(4b) \ ; \ (0 \ 1) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad (0 \ 0) = X$$

$$O(2,1) = I(1,1) = I = \Rightarrow$$

$$s(1a) \underline{I} = s(2b) \ ; \ (1 \ 0) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (1 \ 0) = T$$

$$s(2a) \underline{I} = s(3b) \ ; \ (0 \ 1) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \quad (1 \ 1) = D$$

BASICS OF UNARY RELATION

$$\underline{a} \underline{Z} \underline{b} \equiv \underline{b} \underline{Z}^t \underline{a} \quad (\text{Relation}) \quad (1a,b)$$

$$(\underline{a} \underline{Z} = \underline{b}) \equiv (\underline{b} \underline{Z}^t = \underline{a}) \quad (\text{Unary form}) \quad (2a,b)$$

$$\text{e.g. } (\underline{a} \Rightarrow \underline{b}) \equiv (\neg \underline{b} \Rightarrow \neg \underline{a}) \quad (3a,b)$$

$$(1 \ 0) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1 \ 0); (0 \ 1) \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (0 \ 1) \quad (4a)$$

$$(0 \ 1) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1 \ 1); (1 \ 0) \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = (1 \ 1) \quad (4b)$$

$$(1) \equiv (2) \equiv (\underline{a} \underline{Z} \underline{b} = T) \quad (5a) \quad \equiv (\underline{b} \underline{Z}^t \underline{a} = T) \quad (5b)$$

$$\text{Write (5a) as } (\underline{a} (T, \underline{Z}) = \underline{b}) \equiv (\underline{a} \underline{Z} = \underline{b}) \quad (2a) \equiv (\underline{a} \underline{Z} \underline{b} = T) \quad (1a) \quad (6)$$

(Binary reverse) (Unary) (Relation)

Then

$$(\underline{a} (F, \underline{Z}) = \underline{b}) \equiv (\underline{a} \underline{Z} \underline{b} = F) \equiv (\underline{a} \underline{Z}^c \underline{b} = T) \equiv (\underline{a} \underline{Z}^c = \underline{b}) \quad (7)$$

(Binary reverse) (Relation) (Relation) (Unary)

TRANSLATION OF BVMF INTO STANDARD LOGIC(a) UNARY RELATION

$((1) \equiv T) \equiv (2)$ leads to

$$(\underline{a} \underline{Z} \underline{b} = T) \equiv (\underline{a} \underline{Z} = \underline{b}) \quad (8)$$

In standard language, this becomes

$$a' \wedge (a \supset b) \supset b', \quad \text{where} \quad (\underline{a}' \underline{Z} = \underline{b}') \quad (9a,b)$$

For $\underline{Z} = \underline{I}$, (9a) is equivalent to two relations (10a,b) in standard logic

$$a \wedge (a \Rightarrow b) \supset b \quad (10a); \quad \neg a \wedge (a \Rightarrow b) \supset (b \vee \neg b) \quad (10b)$$

and the equivalent BVMF equations for (9b) are

$$\underline{Z} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \mapsto (1 \ 0) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1 \ 0); \quad (0 \ 1) \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = (1 \ 1) \quad \begin{matrix} (11 \\ a,b,c) \end{matrix}$$

$$\text{i.e. } \underline{a}' \underline{I} = \underline{b}' \quad a_T' \mapsto b_T', \quad a_F' \mapsto b_D' \equiv b_T' \vee b_F' \quad (12a,b,c)$$

(9) is the generalization of (10), (11), (12) — e.g

$$\wedge(a \wedge b) \supset b \quad (13a); \quad \neg a \wedge (a \wedge b) \supset (b \wedge \neg b) \quad (\underline{Z} = \underline{A}) \quad (13b)$$

$$\wedge(a \vee b) \supset (b \vee \neg b) \quad (14a); \quad \neg a \wedge (a \vee b) \supset b \quad (\underline{Z} = \underline{0}) \quad (14b)$$

The generalization of (8) to $\underline{a} \underline{Z} \underline{b} = F$ is the

"binary reverse relation"

TRANSLATION OF BVMF INTO STANDARD LOGIC(b) BINARY FORWARD RELATION

$$(\underline{a} \underline{Z} \underline{b} = \underline{c}) \equiv (a \underline{Z} b \underline{=} c) \quad (15a,b)$$

e.g.

$$(\underline{a} \underline{A} \underline{b} = \underline{c}) \equiv ((a \wedge b) \supset c) \wedge (\neg(a \wedge b) \supset \neg c) \quad (16a,b)$$

$$(\underline{a} \underline{O} \underline{b} = \underline{c}) \equiv ((a \vee b) \supset c) \wedge (\neg(a \vee b) \supset \neg c) \quad (17a,b)$$

each

In fact, (16) and (17) are equivalent to two equations in (18a,b) and (19a,b)

$$(a_\alpha \wedge b_\alpha = c_\alpha, a_\beta \vee b_\beta = c_\beta) \equiv (a \wedge b \supset c) \wedge (\neg a \vee \neg b \supset \neg c) \quad (18a,b)$$

$$(a_\alpha \vee b_\alpha = c_\alpha, a_\beta \wedge b_\beta = c_\beta) \equiv (a \vee b \supset c) \vee (\neg a \wedge \neg b \supset \neg c) \quad (19a,b)$$

Eqn (15a) is equivalent to the BVMF equation

$$\langle a | z | b \rangle = c_\alpha, \quad \langle a | z^c | b \rangle = c_\beta, \quad (c_\alpha \ c_\beta) = \langle c | \quad (20)$$

TRANSLATION OF BVMF INTO STANDARD LOGIC(c) BINARY REVERSE RELATION

We repeat (8) and (7)

$$(\underline{a} \underline{Z} \underline{b} = T) \equiv (\underline{a} \underline{Z} = \underline{b}) \equiv (\underline{b} \underline{Z}^T = \underline{a}) \quad (21a,b,c)$$

$$(\underline{a} \underline{Z} \underline{b} = F) \equiv (\underline{a} \underline{Z}^C \underline{b} = T) \equiv (\underline{a} \underline{Z}^C = \underline{b}) \equiv (\underline{b} \underline{Z}^{ct} = \underline{a}) \quad (22a,b,c)$$

Write (21a), (22a) as

$$\underline{a} (\underline{c}, \underline{Z}) = \underline{b}, \text{ for } \underline{c} = T, F \quad (23)$$

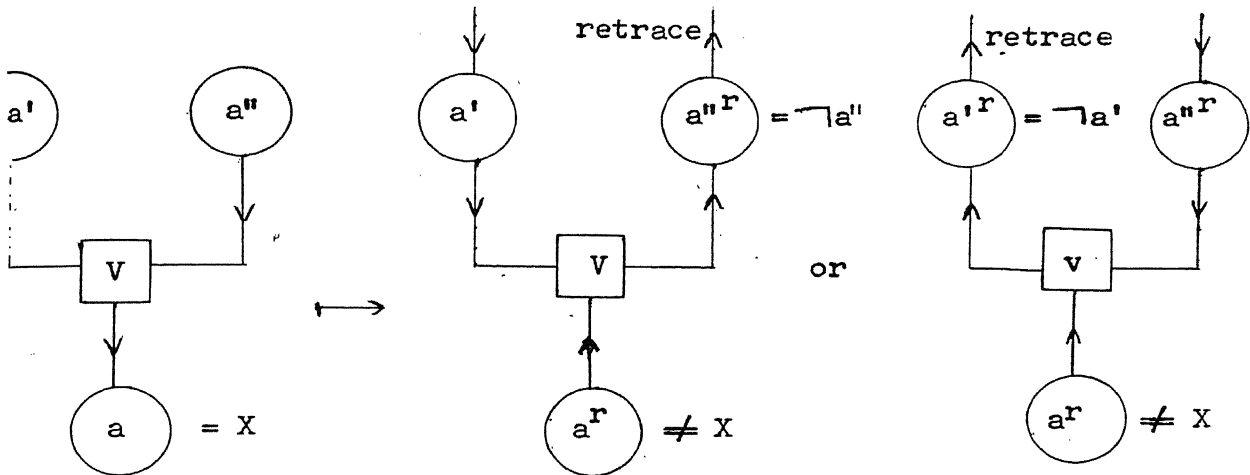
and generalize to $\underline{c} = (D = T \oplus F)$ and $(X = T \otimes F)$ by

$$(\underline{a} (D, \underline{Z}) = \underline{b}) \equiv \underline{a}(\underline{Z} \oplus \underline{Z}^C) = \underline{a} \underline{D} = \underline{b} = D ; \quad \left| \underline{D} \right| = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad (24a)$$

and

$$(\underline{a} (X, \underline{Z}) = \underline{b}) \equiv \underline{a}(\underline{Z} \otimes \underline{Z}^C) = \underline{a} \underline{X} = X ; \quad \left| \underline{X} \right| = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad (24b)$$

.A5.



e.g. $a' = T, a'' = F \mapsto a = T \wedge F = (1 \ 0) \otimes (0 \ 1) = (0 \ 0) = X$

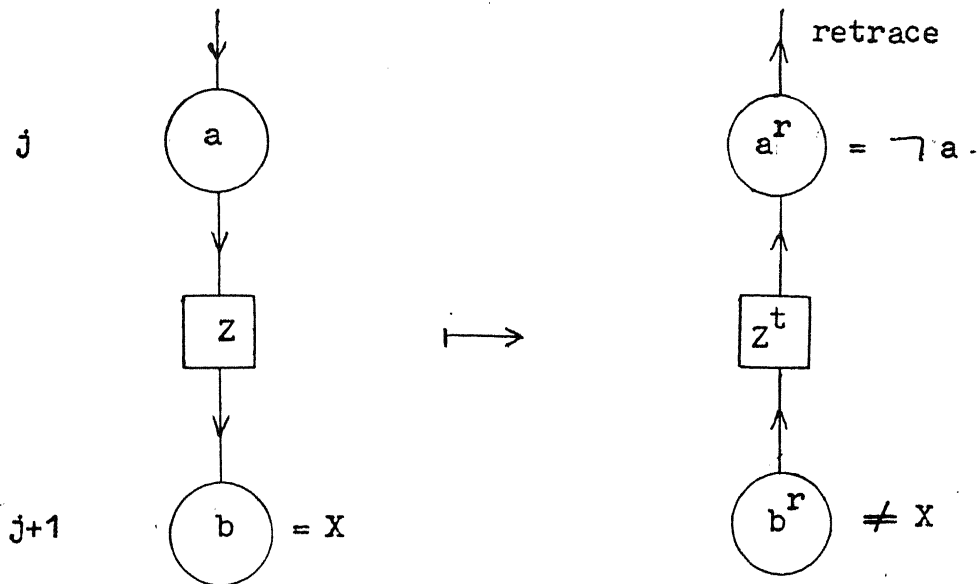
$a' = T, a''^r = T \mapsto a = T \neq X$; Retrace from a''^r

or

$a'^r = F, a'' = F \mapsto a = F \neq X$; Retrace from a'^r

(a) Vidya check (Boolean connective)

Fig.2 The only three possible ways of a contradiction arising in a logical graph.



$$s(k^c a) \underline{A}(k, \ell) = s(\ell b) = X ;$$

$$\text{e.g. } \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} / \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

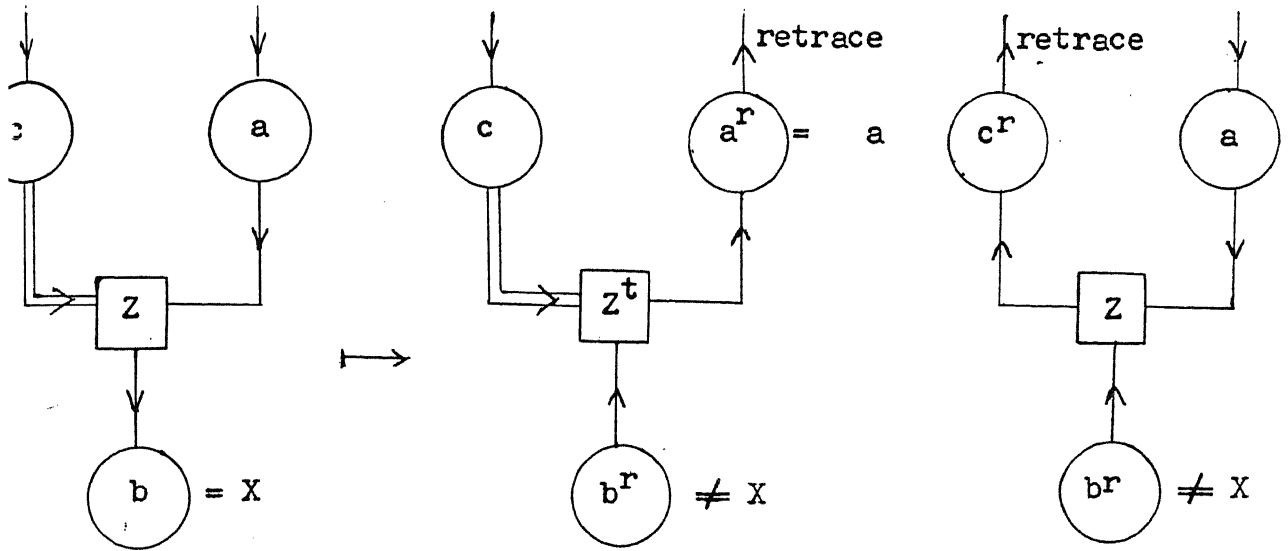
$\underline{O}(k, \ell)$ and $\underline{E}(k, \ell)$
cannot yield $\underline{b} = X$

$$s(ka) \underline{A}(k, \ell) = s(\ell b) \neq X \quad (\ell = 1 \text{ or } 2)$$

$$\text{e.g. } \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

Hence, $\underline{b}^r \neq X \mapsto \underline{a}^r = \neg \underline{a} \quad (k^c a \quad ka)$
Retrace from \underline{a}^r

Fig.2(b) Unary relation (Matrix connective)



Same as for (b)

See below

$$s(k^c a) (T, \underline{A}(k, l)) \equiv s(k^c a) \underline{A}(k, l) = X$$

$$s(k^c a) (F, \underline{A}(k, l)) \equiv s(k^c a) \underline{Q}(k^c, l^c) = D$$

$$\text{e.g. } s(1a) \underline{A}(2,1) \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} ; s(1a) \underline{Q}(1,2) \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

$$s(k^c a) (F, \underline{Q}(k^c, l^c)) \equiv s(k^c a) \underline{A}(k, l) = X$$

$$s(k^c a) (T, \underline{Q}(k^c, l^c)) \equiv s(k^c a) \underline{Q}(k^c, l^c) = D$$

In either case, retrace from $\underline{c}^r = \neg \underline{c}$

Fig.2(c) Binary reverse relation (Matrix connective)

Explanation of Fig. 3(a) — Reversal of unary relation

To show $\underline{a} \underline{Z} = \underline{b}$, $\underline{b}^r = \neg \underline{b} \mapsto \underline{a}^r = \neg \underline{a}$ for
 $\underline{Z} = \underline{O}(I)$, \underline{E} -type and X for $\underline{Z} = \underline{A}$ -type

For \underline{b}^r to go through, \underline{b} must be T or F . If $\underline{b} = D$
 $\underline{b}^r = \neg \underline{b} = D = \underline{b}$ and the reversal stops there. We
 are also considering the case of $\underline{b} \neq X$.

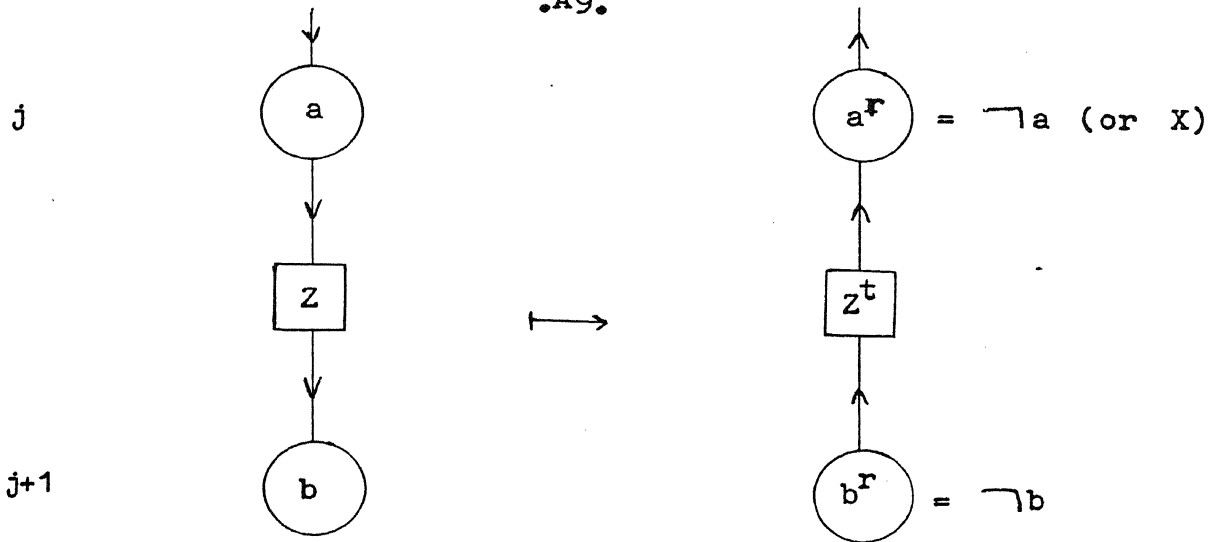
O-type

$s(k^c a) \underline{O}(k, \ell)$ $s(\ell b)$, replaced by $s(\ell^c b^r)$ Forward
 $s(\ell^c b^r) \underline{O}(\ell, k) \mapsto s(ka^r)$, as against $s(k^c a)$ Reverse
 i.e. $(\underline{a} \underline{O} = \underline{b} \text{ and } \underline{b} \neq D) \mapsto (\neg \underline{b} \underline{O}^t = \neg \underline{a})$

A-type

$s(ka) \underline{A}(k, \ell) \mapsto s(\ell b)$, replaced by $s(\ell^c b^r)$ Forward
 $s(\ell^c b^r) \underline{A}(\ell, k) \mapsto s(4a^r) = X$ Reverse
 i.e. $(\underline{a} \underline{A} = \underline{b} \text{ and } \underline{b} \neq X) \mapsto (\neg \underline{b} \underline{A}^t = X)$

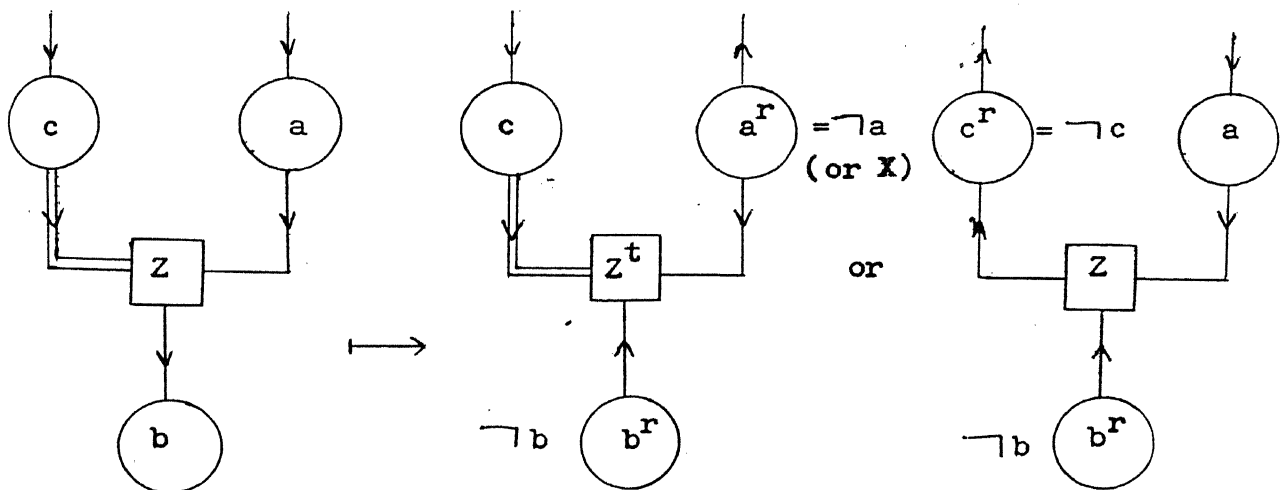
.A9.



$\underline{a}^r = \underline{a}$ for $\underline{Z} = \underline{E}$ or $\underline{O}(\underline{I})$ -type

$\underline{a}^r = X$ for $\underline{Z} = \underline{A}$ -type

(a) Unary matrix relation



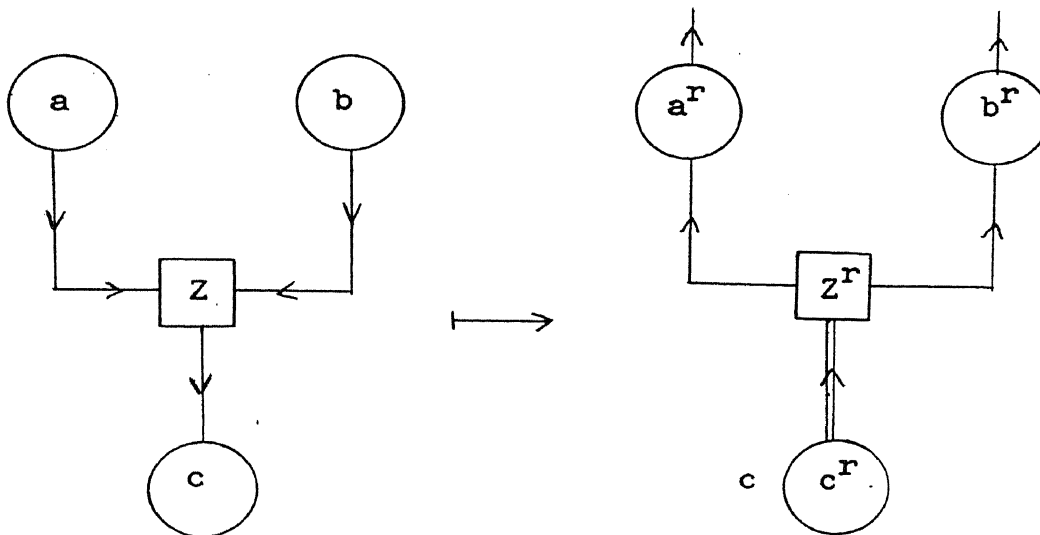
Same as for (a)

$\underline{c}^r = \neg \underline{c}$ for

$\underline{Z} = \underline{E}, \underline{O}(\underline{I}), \underline{A}$

(b) Binary reverse matrix relation

Fig.3 Implementation in the reverse direction (back-tracking) of matrix connective relations.



$$\underline{Z} = \underline{A}(k, \ell) \mapsto \underline{Z}^r = \underline{O}(k^c, \ell^c) \quad \underline{k}a^r = \underline{k}^c \text{ OR } \ell b^r = \ell$$

$$\underline{Z} = \underline{O}(k, \ell) \mapsto \underline{Z}^r = \underline{A}(k^c, \ell^c) \quad \underline{k}a^r = \underline{k}^c \text{ AND } \ell b^r = \ell$$

$$\underline{Z} = \underline{E}(k, \ell) \mapsto \underline{Z}^r = \underline{E}^c(k, \ell) \quad \underline{k}a^r = \underline{k}^c \text{ XOR } \ell b^r = \ell$$

(c) Binary forward matrix relation.

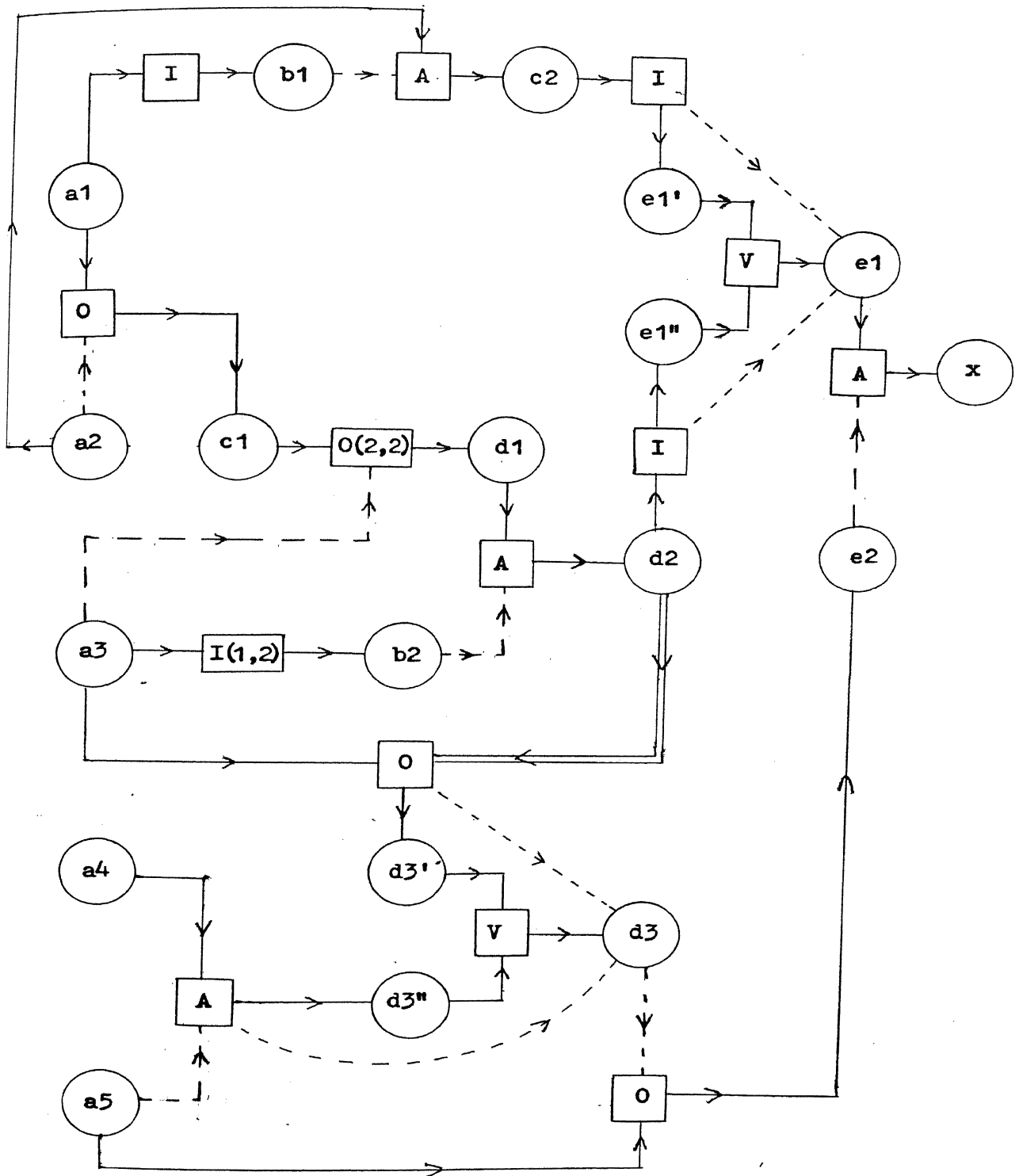


Fig.4 Logical graph of the argument listed in Table 1 .
The dotted lines correspond to two inputs for the terms d3 and e which are compared by the operator V in each case before implementation.

Table 1. Analysis of the listing of a logical argument into its stages 1 to 7

Inputs a1, a2, a3, a4, a5

Sl No	Listing in arbitrary order	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
1	<u>a1</u> <u>I</u> = <u>b1</u>	1, <u>b1</u>						
2	<u>a1</u> <u>O</u> <u>a2</u> = <u>c1</u>	2, <u>c1</u>						
3	<u>a2</u> <u>A</u> <u>b1</u> = <u>c2</u>	3, Hold	3, <u>c2</u>					
4	<u>c1</u> <u>O</u> (2,2) <u>a3</u> = <u>d1</u>	4, Hold	4, <u>d1</u>					
5	<u>a3</u> <u>I</u> (1,2) = <u>b2</u>	5, <u>b2</u>						
6	<u>d1</u> <u>A</u> <u>b2</u> = <u>d2</u>	—	6, Hold	6, <u>d2</u>				
7	<u>a3</u> (<u>d2</u> , <u>0</u>) = <u>d3'</u>	7, Hold	7, Hold	7, Hold	7, <u>d3'</u>			
8	<u>a4</u> <u>A</u> <u>a5</u> = <u>d3''</u>	8, <u>d3''</u>						
8a	<u>d3'</u> <u>V</u> <u>d3''</u> = <u>d3</u>	—	—	—	—	8a, <u>d3</u>		
9	<u>c2</u> <u>I</u> = <u>e1'</u>	—	—	9, <u>e1'</u>				
10	<u>d2</u> <u>I</u> = <u>e1''</u>	—	—	—	10, <u>e1''</u>			
10a	<u>e1'</u> <u>V</u> <u>e1''</u> = <u>e1</u>	—	—	—	10a, Hold	10a, <u>e1</u>		
11	<u>a5</u> <u>O</u> <u>d3</u> = <u>e2</u>	11, Hold	11, Hold	11, Hold	11, Hold	11, Hold	11, <u>e2</u>	
12	<u>e1</u> <u>A</u> <u>e2</u> = <u>x</u>	—	—	—	—	—	12, Hold	12, <u>x</u>

Table 2. Rearranging the arbitrary listing of Table 1 into sequential order

Inputs

a1 , a2 , a3 , a4 , a5

Stage 1

a1 I = b1 (1.1); a1 O a2 = c1 (1.2); a3 I(1,2) = b2 (1.3) ;
a4 A a5 = d3" (1.4)

Stage 2

a2 A b1 = c2 (2.1) ; c1 O(2,2) a3 = d1 (2.2)

Stage 3

d1 A b2 = d2 (3.1) ; c2 I = e1' (3.2)

Stage 4

a3 (d2, O) = d3' (4.1) ; d2 I = e1" (4.2)

Stage 5

d3' V d3" = d3 (5.1) ; e1' V e1" = e1 (5.2)

Stage 6

a5 O d3 = e2 (6)

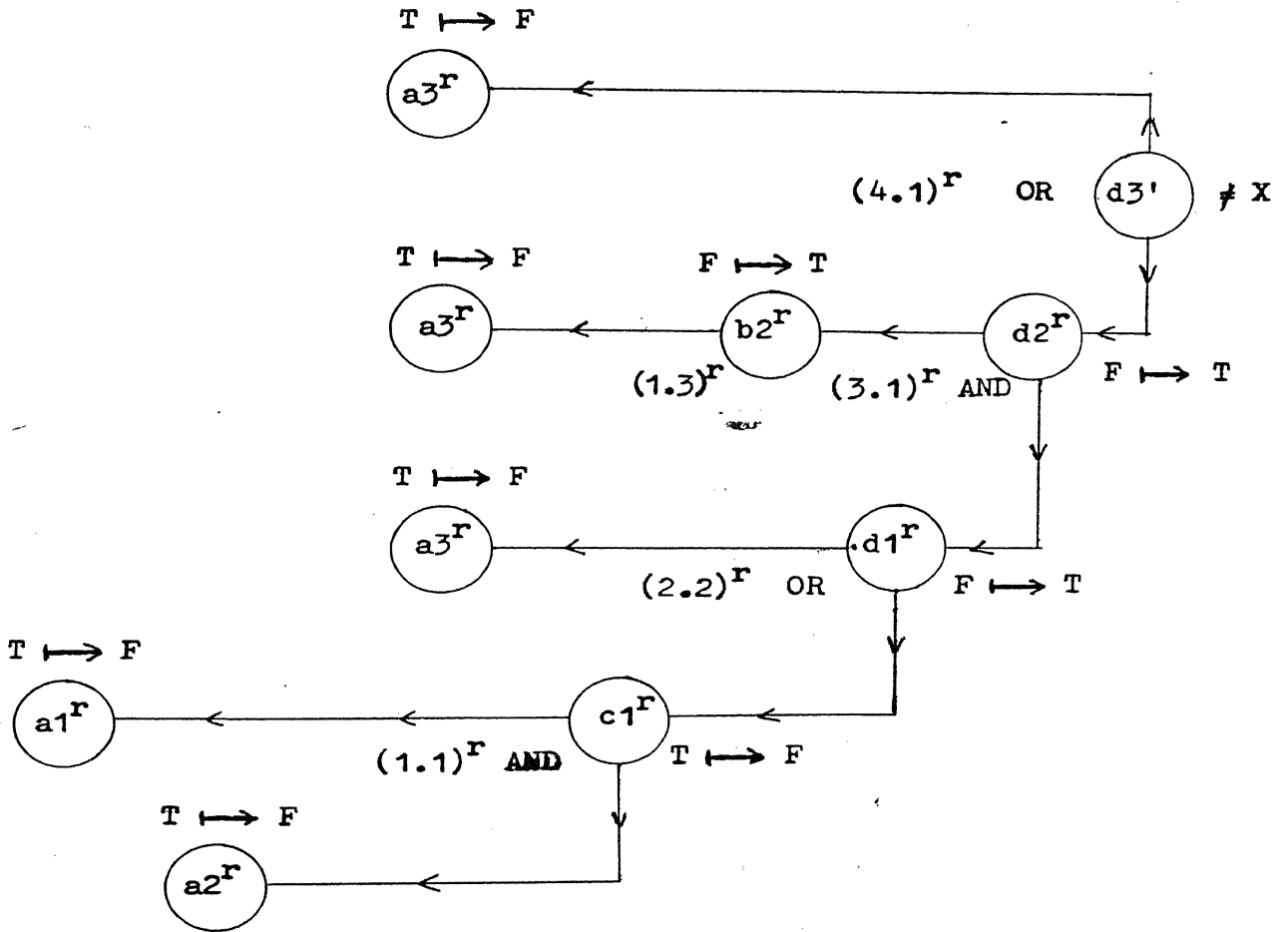
Stage 7

e1 A e2 = x (7)

Table 3 Implementation of the logical graph in Fig. 4 for
given inputs

Sl No	Input					Intermediate input-outputs for stages										Output state			
	a1	a2	a3	a4	a5	1				2		3		4		5		6	7
						b1	c1	b2	d3''	c2	d1	d2	e1'	d3'	e1''	d3	e1	e2	
1	T	T	T	T	T	T	T	F	T	T	F	F	T	X	D	-	-	-	-
2	T	T	F	T	T	T	T	D	T	T	T	D	T	D	D	T	T	T	T
3	F	F	F	T	T	D	F	D	T	D	T	D	D	D	D	T	D	D	D
4	F	F	T	T	T	D	F	F	T	F	T	F	D	X	D	-	-	-	-

.A15.



Revision needed : $(a3^r \vee (a3^r \wedge (a3^r \vee (a1^r \wedge a2^r))))$
 $= (a3^r) \vee (a3^r \wedge a2^r \wedge a1^r)$

Fig.5. Flow chart of the backtracking from the contradiction at $d3'$ for the forward implementation of Fig.4 with the inputs in Row 1 of Table 3.

BOOLEAN VECTOR-MATRIX FORMULATION (BVMF) OF LOGIC

Lecture 3 and 4 , Series 2

Lecture 3 : Essentials of quantifier logic

Lecture 4 : Details of QL-1 and QL-2 algebra

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

BOOLEAN VECTOR-MATRIX FORMULATION (BVMF) OF LOGIC

Lectures 3 and 4 , Series 2

Lecture 3 : Essentials of quantifier logic

Lecture 4 : Details of QL-1 and QL-2 algebra

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Preface

This report contains the essential ideas concerned with the formulation of quantifier states in BVMF and the application of these for the two types of quantified statements in predicate calculus, namely those belonging to QL-1 and QL-2. The material contained in this report is in the form of overhead projector formulae and detailed explanations are not given, as these lectures will be revised and rewritten in Series-3 for application. So also, some of the formulae, as in page 1, Lecture-4, require emendation although they are basically correct, since the definition of relative truth values in SNS logic and quantifier logic have been more precisely given in Lecture Series-3, they are left as they are, since the definitions given here are employed in the succeeding pages of Lecture-4.

Lecture 3 : Essentials of quantifier logic

DEFINITION OF THE FOUR CLASSICAL QUANTIFIERS

Subset $A = \{A_i\}$ of universal set $S \equiv \{A_1, A_2, \dots, A_n\}$
with n members is described by BVMF vector \mathcal{Q} (1)

Define $\mathcal{Q} = (a_1, a_2, \dots, a_i, \dots, a_n)$;
 $a_i = 1$, if $A_i \in A$ and 0 , if $A_i \notin A$ (2)

Define $\underline{q} = (q_i)(a_i) =$ quantifier state of A (3)

The four classical quantifiers are:

$q = \forall$: "For all" ; $q = \exists$: "There exists" ; (4a)

$q = \Phi$: "For none" ; $q = \neg$: "Not for all"

Definition

$$\forall : (\forall i) (a_i) \iff \bigwedge_i a_i = 1 \quad (5)$$

$$\exists : (\exists i) (a_i) \iff \bigvee_i a_i = 1 \quad (6)$$

$$\Phi : (\forall i) (\neg a_i) \iff \bigwedge_i (\neg a_i) = 1 \quad (\equiv \bigwedge_i a_i^c = 1) \quad (7)$$

$$\neg : (\exists i) (\neg a_i) \iff \bigvee_i (\neg a_i) = 1 \quad (\equiv \bigvee_i a_i^c = 1) \quad (8)$$

ADDITIONAL QUANTIFIERS PRODUCED BY BOOLEAN CONNECTIVES

Just as two additional states $D = T \oplus F$, $X = T \otimes F$
in SNS logic, four more quantifier states, named

$$\begin{aligned} q = \Delta & : \text{"Indefinite"} & ; & \quad q = \emptyset : \text{"Impossible" or "null set"} \\ q = \Sigma & : \text{"For some"} & ; & \quad q = \ominus : \text{"For all or none"} \end{aligned} \quad (4b)$$

are definable as in Eqs (9) to (12) :

$$\begin{aligned} \Delta : (\Delta i)(a_i) &= (\exists i)(a_i) \oplus \neg(\exists i)(a_i) \\ &\iff (\exists i)(a_i) \oplus (\forall i)(\neg a_i) \end{aligned} \quad (9)$$

$$\begin{aligned} \emptyset : (\emptyset i)(a_i) &= (\forall i)(a_i) \otimes \neg(\forall i)(a_i) \\ &\iff (\forall i)(a_i) \otimes (\exists i)(\neg a_i) \end{aligned} \quad (10)$$

$$\Sigma : (\Sigma i)(a_i) = (\exists i)(a_i) \otimes (\exists i)(\neg a_i) \quad (11)$$

$$\ominus : (\ominus i)(a_i) = (\forall i)(a_i) \oplus (\forall i)(\neg a_i) \quad (12)$$

The connectives \oplus and \otimes are different from \vee and \wedge in standard logic (although they are equivalent to one another in BA-1) and correspond to the logical relations

$$\vee \underline{a'} \quad \underline{a''} \quad \text{and} \quad \wedge \underline{a'} \quad \underline{a''} \quad (13)$$

where $\underline{a'}$ and $\underline{a''}$ are information from two sources regarding the quantifiers state of the same term \underline{a} .

DESCRIPTION OF THE EIGHT QUANTIFIERS WITH BA-1 TRUTH VALUES

The number (\mathcal{V}) of members A_i that are present in the subset \mathcal{A} with quantifier state $(q_i)(a_i)$ are as follows:

Table 1 Semi-intuitive derivation of generators of QBA

Quantifier state $(q_i)(a_i)$	Description in Set Theory	QBA symbol	Range of \mathcal{V}	QBA description
<u>Standard quantifiers</u>				
$(\forall i)(a_i)$	All A_i are present	\forall	$\mathcal{V} = n$	\forall
$(\exists i)(a_i)$	Some or all A_i are present	\exists	$\mathcal{V} = 1$ to n	$\forall \oplus \Sigma$
$(\exists i)(\neg a_i)$	Not all A_i are present	$\neg \forall$	$\mathcal{V} = 0$ to $(n-1)$	$\Phi \oplus \Sigma$
$(\forall i)(\neg a_i)$	No A_i are present	Φ	$\mathcal{V} = 0$	Φ
<u>Additional quantifiers</u>				
$(\Sigma i)(a_i)$	Some A_i are present, but not all and not none	Σ	$\mathcal{V} = 1$ to $(n-1)$	Σ
$(\ominus i)(a_i)$	All A_i or no A_i are present	\ominus	$\mathcal{V} = 0$ or $\mathcal{V} = n$	$\forall \oplus \Phi$
$(\Delta i)(a_i)$	All A_i , or no A_i , or some A_i are present	Δ	$\mathcal{V} = 0$ to n	$\Sigma \oplus \forall \oplus \Phi$
$(\phi i)(a_i)$	None of the above seven possibilities	ϕ	—	$\Sigma \otimes \forall \otimes \Phi$

The ranges of \mathcal{V} for the eight possible quantifiers are describable as Boolean sums of the ranges for \forall , Σ , Φ shown in Fig.1. The eight ranges thus form a BA-3 Boolean algebra, named Quantifier Boolean Algebra (QBA) whose generators are indicated in Fig.1.

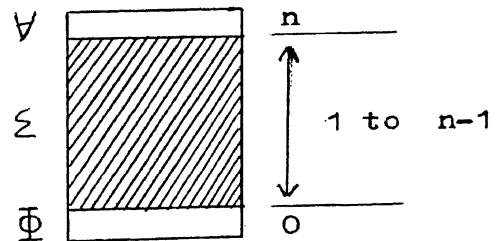


Fig.1. Ranges of \mathcal{V} for the generators \forall , Σ and Φ of the BA-3 algebra of QBA (3)

QUANTIFIER BOOLEAN ALGEBRA AND ITS EIGHT STATES

Table 2. BA-3 vectors corresponding to the eight QBA states
q(1) to q(8)

S1 No	Symbol \underline{q}	Boolean vector of \underline{a} ($a_r a_g a_e$)	QBA description	Classical description
q(1)	\forall	(1 0 0)	Gen*	\forall
q(3)	\exists	(0 1 0)	Gen	$\exists \oplus \exists \neg$
q(5)	Φ	(0 0 1)	Gen	$\forall \neg$
q(2)	$\neg \forall$	(0 1 1)	$\neg \forall$	$\neg \forall$
q(4)	$\neg \exists$	(1 0 1)	$\neg \exists$	$\forall \oplus \forall \neg$
q(6)	$\neg \Phi$	(1 1 0)	$\neg \Phi$	$\neg \forall \neg$
q(7)	Δ	(1 1 1)	$\underline{q} \oplus \neg \underline{q}$	$\forall \oplus \exists \neg, \exists \oplus \forall \neg$
q(8)	\emptyset	(0 0 0)	$\underline{q} \oplus \neg \underline{q}$	$\forall \oplus \exists \neg, \exists \oplus \forall \neg$

*Gen = Generator

Quantified term: $\underline{ax} = (\underline{qx})(\underline{ax})$, (Note use of x, instead of i)

The eight quantifier states in Table 2 are designated q(1) to q(8) (analogous to s(1) to s(4) of SNS truth values T, F, D, X).

The numbering satisfies the condition that

$$q(2n) = q^c(2n-1) = \neg q(2n-1) \quad (\text{complement})$$

q(1), q(3), q(5) are the generators, with one 1 and two 0's in their Boolean vectors.

q(2), q(4), q(6) are the complements of these, with one 0 and two 1's in their Boolean vectors.

q(7) is the indefinite state with three 1's in the Boolean vector.

q(8) is the impossible state with three 0's in its Boolean vector.

STANDARD QUANTIFIERS IN SNS LOGIC

Subset $\mathcal{A} = \{\underline{A}_i\}$ of set $\mathcal{S} = \{\underline{A}_1, \dots, \underline{A}_i, \dots, \underline{A}_n\}$ (14a)

is represented by BVMF (n,2)-vector

$\mathcal{Q} = (\underline{a}_1, \dots, \underline{a}_i, \dots, \underline{a}_n)$, with $\underline{a}_i = (a_{i\alpha} \ a_{i\beta}) = T, F, D, X$ (14b)

corresponding to

\underline{A}_i = present, absent, unknown, contradictory (14c)

We shall use the terms "component \underline{a}_i " interchangeably to stand for the member \underline{A}_i , with index i , having the state of existence corresponding to the SNS truth value of \underline{a}_i .

The four standard quantifier states in SNS

$$\forall : (\forall i)(\underline{a}_i) \iff \bigwedge_i \underline{a}_i = T \quad (15)$$

$$\exists : (\exists i)(\underline{a}_i) \iff \bigvee_i \underline{a}_i = T \quad (16)$$

$$\Phi : (\forall i)(\neg \underline{a}_i) \iff \bigwedge_i (\underline{a}_i \ N) = T \quad (17)$$

$$\Lambda : (\exists i)(\neg \underline{a}_i) \iff \bigvee_i (\underline{a}_i \ N) = T \quad (18)$$

(\bigwedge_i = "multiple and", \bigvee_i = "multiple or", in SNS)

In words

$$\forall : \begin{array}{l} \text{"Everyone of the members} \\ \underline{A}_i \text{ is present"} \end{array} \equiv \text{"For all } \underline{a}_i \text{"} \quad (19)$$

$$\exists : \begin{array}{l} \text{"At least one member} \\ \underline{A}_i \text{ is present"} \end{array} \equiv \text{"There exists } \underline{a}_i \text{"} \quad (20)$$

$$\Phi : \begin{array}{l} \text{"Everyone of the members} \\ \underline{A}_i \text{ is absent"} \end{array} \equiv \text{"For no } \underline{a}_i \text{"} \quad (21)$$

$$\Lambda : \begin{array}{l} \text{"At least one member} \\ \underline{A}_i \text{ is absent"} \end{array} \equiv \text{"Not for all } \underline{a}_i \text{"} \quad (22)$$

REPRESENTATION OF QBA STATES VIA BA-2 TRUTH VALUES

It is possible to formulate quantifier states as well-formed formulae in SNS logic which are representable by expressions in BA-2 algebra.*

Generators are defined as follows

$$(\forall i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = T \quad (\text{All } n \text{ } \underline{a_i} \text{ are } T) \quad (23)$$

$$(\sum i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = D, \text{ but not } T \text{ and not } F. \quad (\text{At least one } \underline{a_i} \text{ is } T \text{ and one } \underline{a_i} \text{ is } F) \quad (24)$$

$$(\bigoplus i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = F \quad (\text{All } \underline{a_i} \text{ are } F) \quad (25)$$

Then, the other five states of QBA have the descriptions in BA-2 algebra as follows.

$$(\bigwedge i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = D, \text{ but not } T \quad (1 \text{ to } n \text{ } \underline{a_i} \text{ are } F \text{ and the rest are } D \text{ or } T) \quad (26)$$

$$(\ominus i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = T \text{ or } F, \text{ but not } D \quad (\text{All } n \text{ } \underline{a_i} \text{ are } T \text{ or all } n \text{ } \underline{a_i} \text{ are } F) \quad (27)$$

$$(\exists i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = D, \text{ but not } F \quad (1 \text{ to } n \text{ } \underline{a_i} \text{ are } T, \text{ and the rest are } D \text{ or } F) \quad (28)$$

$$(\triangle i)(\underline{a_i}) = \bigoplus_i \underline{a_i} = D \quad (\text{All } \underline{a_i} \text{ can be } T, F \text{ or } D) \quad (29)$$

$$(\phi i)(\underline{a_i}) = \bigotimes_i \underline{a_i} = X \quad (1 \text{ to } n \text{ } \underline{a_i} \text{ are } X - \text{ not } T, \text{ not } F \text{ and not } D) \quad (30)$$

* Since the definition of the four non-standard quantifiers needs the Boolean operators \bigoplus and \bigotimes , their definition requires the "multiple sum" $\bigoplus_i \underline{a_i}$.

Table 3: Properties of the quantifier Boolean algebra with
SNS truth values

Sl. No.	Quantifier state q			Truth value of $\bigoplus_x \underline{a}x = B(q)^*$		
	Name	Symbol	Set-theoretical description	Description in BA-2	Symbol in BA-3 [†]	3-vector
q(1)	For all	\forall	All $n \underline{a}x$ are T.	Always T	TT (Gen)	(1 0 0)
q(3)	For some	\exists	1 to $n-1 \underline{a}x$ are T, 1 to $n-1 \underline{a}x$ are F, and the rest are D.	D, but not T and not F	SS (Gen)	(0 1 0)
q(5)	For none	\emptyset	All $n \underline{a}x$ are F.	Always F	FF (Gen)	(0 0 1)
q(6)	There exists	\exists	1 to $n \underline{a}x$ are T and the rest are D or F.	D or T, but not F	ET = $SS \oplus TT$	(1 1 0)
q(4)	All or none	\ominus	All $n \underline{a}x$ are T or all $n \underline{a}x$ are F.	T or F, but not D	TF = $TT \oplus FF$	(1 0 1)
q(2)	Not for all	\wedge	1 to $n \underline{a}x$ are F, and the rest are D or T.	D or F, but not T	EF = $SS \oplus FF$	(0 1 1)
q(7)	Indefinite	Δ	1 to $n \underline{a}x$ may be T, F, or D.	D, may also be T or F	DD = $TT \oplus SS \oplus FF$	(1 1 1)
q(8)	Impossible	ϕ	Null set	Not D, not T and not F	XX = $TT \otimes SS \otimes FF$	(0 0 0)

$\underline{a}x$ is the logical representation of the set-theoretical description. Although its description is in BA-2, its range is not a truth value of BA-2 and requires BA-3 algebra.

Gen = generator.

INTER-RELATIONSHIPS BETWEEN THE STANDARD QUANTIFIERS

From the De Morgan relations in SNS

$$\neg(\underline{a} \underline{O} \underline{b}) = (\underline{a} \underline{N}) \underline{A} (\underline{b} \underline{N}) ; \neg(\underline{a} \underline{A} \underline{b}) = (\underline{a} \underline{N}) \underline{O} (\underline{b} \underline{N}) \quad (32a,b)$$

it follows that

$$\neg(\bigvee_x \underline{ax}) = \bigwedge_x (\underline{ax} \underline{N}) ; \neg(\bigwedge_x \underline{ax}) = \bigvee_x (\underline{ax} \underline{N}) \quad (33a,b)$$

These are well-formed BA-2 formulae of SNS, and they lead to the formulae^(34a,b) connecting the classical quantifiers \forall and \exists , and hence to (34c,d).

$$\neg(\forall x)(\underline{ax}) = (\exists x)(\neg \underline{ax}) ; \neg(\exists x)(\underline{ax}) = (\forall x)(\neg \underline{ax}) \quad (34a,b)$$

$$\neg(\forall x)(\neg \underline{ax}) = (\exists x)(\underline{ax}) ; \neg(\exists x)(\neg \underline{ax}) = (\forall x)(\underline{ax}) \quad (34c,d)$$

In these, the negational operators \neg and \neg are distinguished by their acting respectively on the quantifier and the predicate respectively, and they correspond to the unary operators "complement" and "negation" (\underline{M} and \underline{N}) which are represented by different 3x3 matrices (see next sheet).

We thus obtain a complete definition of the quantifier states in BA-2, which leads to their essential properties required to formulate the algebra of quantifier logic (QL) in BA-3 analogous to the BA-2 algebra of SNS.

NOTE ON THE STATES Δ and ϕ

These are describable in terms of \forall and \exists as follows:

$$(\Delta x)(\underline{ax}) = (\forall x)(\underline{ax} = D) ; (\phi x)(\underline{ax}) = (\exists x)(\underline{ax} = X) \quad (35a,b)$$

INTUITIVE EXTENSION OF QUANTIFIERS TO INFINITE SETS

All the definitions and formulae in BA-1, BA-2 and BA-3 are extendable to infinite sets represented by the n-vector

$$\mathcal{Q} = (\underline{a}_1, \dots, \underline{a}_x, \dots, \underline{a}_n) \text{ for } n \mapsto \infty \quad (36)$$

Although there is no last number n when $n \mapsto \infty$, we define $(\forall x)$ to correspond to the range $\mathcal{R}(\mathcal{V})$ for $\mathbb{L}t(\mathcal{V} = n)$.
 $n \mapsto \infty$

In this way, for a finite interval of rationals (say 0 to 1), defined by

$$x = m/n, \quad m = 1 \text{ to } n, \quad n \mapsto \infty, \quad \underline{a}_x = \underline{a}(m/n) \quad (37)$$

The definition of the quantifiers represented by \mathcal{Q} in (36) still holds for the infinite sets. Only Fig.1 gets modified as Fig.2.

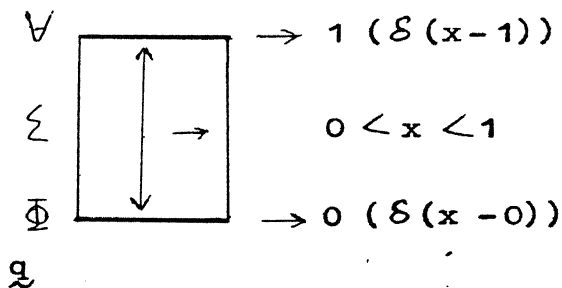


Fig.2. Ranges of x for the generators \forall, Σ, Φ , of QBA in general.

Table 3 of sheet 7 continues to hold for all the eight QBA states.

We shall only use x hereafter for the variable used for labelling the scope of the quantifier in $\underline{a}_x = (\underline{q}_x)(\underline{a}_x)$, irrespective of whether the relevant set is finite or infinite.

EXAMPLES OF QUANTIFIED TERMS AND RELATIONS IN QBA STATES

Suppose x ranges over the integers 1 to n (n finite or $n \mapsto \infty$). Let $\underline{i}_x, \underline{e}_x, \underline{o}_x, \underline{p}_x$ stand for $x = \text{integer, even number, odd number and prime}$. Then the following quantified terms hold with the formula for the predicate being an SNS term.

$$\begin{aligned} &(\forall x)(\underline{i}_x > 0) ; (\Phi x)(\underline{e}_x \Rightarrow \underline{o}_x) ; (\Sigma x)(\underline{p}_x) \\ &(\exists x)(\underline{i}_x \Rightarrow \underline{p}_x) \otimes (\exists x)(\neg(\underline{i}_x \Rightarrow \underline{p}_x)) \equiv (\Sigma x)(\underline{i}_x \Rightarrow \underline{p}_x) ; \\ &(\phi x)(\underline{o}_x \wedge (\underline{o}_x \Rightarrow \underline{e}_x)) \end{aligned}$$

UNARY (MATRIX-cum-BOOLEAN) OPERATORS FOR QUANTIFIED TERMS

For $\underline{a} \equiv (\underline{q}x)(\underline{a}x) \equiv q(\underline{a}x)$, $\underline{a} \underline{B} = \underline{b}$, for the matrix-cum-Boolean operator $\underline{B} = \underline{E}, \underline{M}, \underline{N}, \underline{L}$. The four matrices and the classical and Boolean algebraic equations are given in Table 4.

Table 4. Description and formulae for unary operators in QL

Name	Operator	Matrix	BVMF description	Classical description
Equivalence ($\underline{a}^e = \underline{b}$)	\underline{E}	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\langle q(\underline{a}x) \underline{E} = \langle q(\underline{b}y) $ $j = i^e$	$(\underline{q}x)(\underline{a}x)$
Complement ($\underline{a}^c = \underline{b}$)	\underline{M}	$\begin{pmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{pmatrix}$	$\langle q(\underline{a}x) \underline{M} = \langle q(\underline{b}y) $ $j = i^c$	$\neg(\underline{q}x)(\underline{a}x)$
Negation ($\underline{a}^n = \underline{b}$)	\underline{N}	$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	$\langle q(\underline{a}x) \underline{N} = \langle q(\underline{b}y) $ $j = i^n$	$(\underline{q}x)(\neg \underline{a}x)$
Ellation ($\underline{a}^l = \underline{b}$)	\underline{L}	$\begin{pmatrix} 0 & 0 & c \\ 0 & c & 0 \\ c & 0 & 0 \end{pmatrix}$	$\langle q(\underline{a}x) \underline{L} = \langle q(\underline{b}y) $ $j = i^l$	$\neg(\underline{q}x)(\neg \underline{a}x)$

$\underline{E}, \underline{M}, \underline{N}, \underline{L}$ ($= \underline{M} \underline{N} = \underline{N} \underline{M}$) form a group of order 4, isomorphous to the crystallographic point group $D_2 = 2 \ 2 \ 2$.

Table 5. Transformations between $q(i)$ by unary operators

	\underline{E}	\underline{M}	\underline{N}	\underline{L}
\forall	\forall	\wedge	\exists	\exists
\wedge	\wedge	\forall	\exists	\exists
\exists	\exists	\exists	\forall	\wedge
\exists	\exists	\exists	\wedge	\forall

	\underline{E}	\underline{M}	\underline{N}	\underline{L}
Σ	Σ	\ominus	Σ	\ominus
\ominus	\ominus	Σ	Σ	\ominus
Δ	Δ	\emptyset	Δ	\emptyset
\emptyset	\emptyset	Δ	\emptyset	Δ

PURE BOOLEAN CONNECTIVES IN QL-1 AND QL-2 ALGEBRAS

Quantifier Logic (QL) can follow two algebras QL-1 and QL-2 for logical operations (both Boolean and matrix) between QBA states according as the quantifier is interpreted in an individual sense or a collective sense. For Boolean operators "Union" (U) and "Vidya" (V), the definition and formulae are as follows:

QL-1 algebra

$$\underline{a}' \underline{U} \underline{a}'' = \underline{a} \equiv \underline{a}' \oplus \underline{a}'' = \underline{a} \equiv q(ia'x) \otimes q(ia''x) = q(iax) \text{ (into)}$$

$$\underline{a}' \underline{V} \underline{a}'' = \underline{a} \equiv \underline{a}' \otimes \underline{a}'' = \underline{a} \equiv q(ia'x) * q(ia''x) = q(iax) \text{ (star)}$$

In this, the Boolean sum and product are applied to the component SNS vectors \underline{a}_i of \underline{a} , and hence the operators are indicated by the SNS format \underline{U} and \underline{V} .

QL-2 algebra

$$\underline{a}' \underline{U} \underline{a}'' = \underline{a} \equiv \left(\bigoplus_i \underline{a}'_i \oplus \bigoplus_i \underline{a}''_i = \bigoplus_i \underline{a}_i \right)$$

$$\equiv (\mathcal{R}'(\mathcal{V}) \cup \mathcal{R}''(\mathcal{V}) = \mathcal{R}(\mathcal{V})) \equiv q(ia'x) \oplus q(ia''x) = q(iax) \text{ (sum)}$$

$$\underline{a}' \underline{V} \underline{a}'' = \underline{a} \equiv \left(\bigoplus_i \underline{a}'_i \otimes \bigoplus_i \underline{a}''_i = \bigoplus_i \underline{a}_i \right)$$

$$\equiv (\mathcal{R}'(\mathcal{V}) \cap \mathcal{R}''(\mathcal{V}) = \mathcal{R}(\mathcal{V})) \equiv q(ia'x) \otimes q(ia''x) = q(iax) \text{ (product)}$$

In this, the BA-3 truth values, describable by the ranges $\mathcal{R}(\mathcal{V})$ of the BA-1 description (sheet 3), of the BA-2 truth values T, F, D, for the values of $\bigoplus_i \underline{a}_i$, defining a quantifier state in a collective manner, are combined by the Boolean operators Union and Vidya. Therefore, the QBA Boolean sum and product operators \underline{U} and \underline{V} are relevant for QL-2.

QL-1 AND QL-2 BOOLEAN OPERATORS

The two operators "sum" (\oplus) and "product" (\otimes) for QL-2 and "into" (\boxplus) and "star" (\star) for QL-1 are describable by the following formulae.

QL-2 OPERATORS

$$\underline{a}' \oplus \underline{a}'' = \underline{a} \equiv \underline{a}'_{\lambda} \oplus \underline{a}''_{\lambda} = \underline{a}_{\lambda} \quad , \quad \lambda = \gamma, \delta, \epsilon$$

$$\underline{a}' \otimes \underline{a}'' = \underline{a} \equiv \underline{a}'_{\lambda} \otimes \underline{a}''_{\lambda} = \underline{a}_{\lambda} \quad , \quad \lambda = \gamma, \delta, \epsilon$$

These are analogous to the corresponding operators in SNS, ^{for \underline{a}} , where $\lambda = \alpha, \beta$.

QL-1 OPERATORS

$$\underline{a}' \boxplus \underline{a}'' = \underline{a}' \oplus \underline{a}'' \quad \text{for } \underline{a}', \underline{a}'' = \text{the three generator states } \underline{V}, \underline{\Sigma}, \underline{\Phi}, \text{ except that } \underline{\Sigma} \boxplus \underline{\Sigma} = \underline{\Sigma} \text{ or } \underline{\Delta}$$

$$\underline{a}' \star \underline{a}'' = \underline{a}' \otimes \underline{a}'' \quad \text{for } \underline{a}', \underline{a}'' = \text{the three generator states } \underline{V}, \underline{\Sigma}, \underline{\Phi}, \text{ except that } \underline{\Sigma} \star \underline{\Sigma} = \underline{\Sigma} \text{ or } \underline{\phi}$$

The former occurs extensively in the algebra of QL-1 matrix operators. The latter is a novel result yet to be evaluated.

For mixed states in QBA ($\underline{a}' = \underline{q}_1 \oplus \underline{q}_2$, $\underline{a}'' = \underline{q}_3 \oplus \underline{q}_4$), all these operators \underline{B} are distributive with respect to the Boolean sum of generator states, and can be worked out as follows.

$$(\underline{q}_1 \oplus \underline{q}_2) \underline{B} (\underline{q}_3 \oplus \underline{q}_4) = (\underline{q}_1 \underline{B} \underline{q}_3) \oplus (\underline{q}_1 \underline{B} \underline{q}_4) \oplus (\underline{q}_2 \underline{B} \underline{q}_3) \oplus (\underline{q}_2 \underline{B} \underline{q}_4)$$

where $\underline{B} = \oplus, \otimes, \boxplus \text{ or } \star$.

QL-1 and QL-2 MATRIX OPERATORS IN PREDICATE LOGICDescription and application to QL-1A and QL-2A types in practice

In first order predicate logic, two types of logical relations are employed, named QL-1A and QL-2A which are implementable in BVMF via QL-1 and QL-2 matrix relations, analogous to the QL-1 and QL-2 Boolean relations discussed above. These have the forms

QL-1 algebra : $\underline{ax} \underline{Z} \underline{bx}$; QL-2 algebra: $\underline{ax} \underline{Z} \underline{bx}$

and the two differ in exactly the same manner as the QL-1 and QL-2 type of Boolean operators differ — namely that the former is operative for every \underline{ax} individually while the latter is operative over the complete quantifier state in a collective manner. The theory of these is given in the next lecture. Here, the application to what has been named QL-1A and QL-2A types which are the standard forms employed in predicate logic are pointed out. The elementary logical relations of the two types are exemplified in standard notation as follows:

QL-1A type: Unary : $(\forall x)(\underline{ax} \implies \underline{bx})$; Binary : $(\exists x)(\underline{ax} \wedge \neg \underline{bx})$

QL-2A type:

Unary: $(\forall x)(\underline{ax}) \implies (\exists y)(\underline{by})$; Binary : $(\exists x)(\underline{ax}) \wedge \neg (\forall x)(\neg \underline{by})$

The BVMF notations for these two types of relations are

QL-1A type: $q(iZx)(\underline{ax} \underline{Z}(k, \ell) \underline{bx}) \equiv \underline{ax} \underline{Z} \underline{bx}$, where $\underline{Z} = (q(iZx) \underline{Z}(k, \ell))$

QL-2A type: $\underline{ax} q(iZx) \underline{Z}(k, \ell) q(jZy) \underline{by} = \underline{ax} \underline{Z}(i, j ; k, \ell) \underline{by}$

The implementation of the former two equations in a practical way is given in the next sheet. The basic theory of QL-1 algebra and QL-2 algebra follow thereafter.

FORM OF IMPLEMENTATION OF QL-1A and QL-2A TYPES

QL-1A type

Logical relation : $q(izx) (\underline{ax} \underline{Z}(k, \ell) \underline{bx})$

Unary implementation

$$q(iax), q(izx) (\underline{ax} \underline{Z}(k, \ell) = \underline{bx}) \mapsto q(ibx)$$

Binary implementation

$$q(iax), q(iby), q(izx) (\underline{ax} \underline{Z}(k, \ell) \underline{bx}) = \underline{c} \mapsto \underline{t}(\underline{Z} \mid \underline{ax}, \underline{bx}) \quad s(kc)$$

QL-2A type

Logical relation : $\underline{ax} \ q(izx) \ \underline{Z}(k, \ell) \ q(jby) \ \underline{by}$

Unary implementation : $q(iax), q(izx) \underline{Z}(k, \ell) = q(jzy) \mapsto q(jby)$

Binary implementation

$$q(iax), q(jby), q(izx) \underline{Z}(k, \ell) \ q(jzy) = \underline{c} \mapsto \underline{t}(\underline{Z} \mid \underline{ax}, \underline{by}) = s(kc)$$

Examples

QL-1A type

Unary : $(\exists x)(\underline{ax}), (\forall x)(\underline{ax} \implies \underline{bx}) = (\exists x)(\underline{bx})$

Binary : $(\forall x)(\underline{ax}), (\exists x)(\underline{bx}), (\exists x)(\underline{ax} \wedge \underline{bx}) \mapsto \underline{t}(\underline{Z} \mid \underline{ax}, \underline{bx}) =$

QL-2A type

Unary : $(\forall x)(\underline{ax}), (\exists x)(\underline{ax}) \implies (\exists y)(\underline{by}) = (\exists y)(\underline{by})$

Binary : $(\forall x)(\underline{ax}), (\exists y)(\underline{by}), (\exists x)(\underline{ax}) \wedge (\exists y)(\underline{by})$
 $\mapsto \underline{t}(\underline{Z} \mid \underline{ax}, \underline{bx}) = T$

QL-1 AND QL-2 MATRIX OPERATORS IN PREDICATE LOGIC

Tables of QL-1A and QL-2A relations

QL-1A type

Unary Relation			Binary relation			
q(iax)	Relation	q(ibx)	q(iax)	q(ibx)	Relation	$\underline{t} = s(kc)$
\forall	$(\forall x)(\underline{a}x \Rightarrow \underline{b}x)$	\forall	\forall	\exists	$(\exists x)(\underline{a}x \vee \underline{b}x)$	T
\exists	$(\forall x)(\underline{a}x \Rightarrow \underline{b}x)$	\exists	\exists	\exists	$(\exists x)(\underline{a}x \vee \underline{b}x)$	D
\forall	$(\exists x)(\underline{a}x \Rightarrow \underline{b}x)$	\exists	\forall	\exists	$(\forall x)(\underline{a}x \vee \underline{b}x)$	T
\exists	$(\exists x)(\underline{a}x \Rightarrow \underline{b}x)$	Δ	\vee	\vee	$(\forall x)(\underline{a}x \vee \underline{b}x)$	D

QL-2A type

Unary relation			Binary relation			
q(iax)	Relation	q(iby)	q(iax)	q(iby)	Relation	$\underline{t} = s(kc)$
\forall	$(\forall x)(\underline{a}x) \Rightarrow (\exists y)(\underline{b}y)$	\exists	\forall	\exists	$(\exists x)(\underline{a}x) \wedge (\exists y)(\underline{b}y)$	T
\exists	$(\forall x)(\underline{a}x) \Rightarrow (\exists y)(\underline{b}y)$	Δ	\exists	\exists	$(\exists x)(\underline{a}x) \wedge (\exists y)(\underline{b}y)$	T
\forall	$(\exists x)(\underline{a}x) \Rightarrow (\exists y)(\underline{b}y)$	\exists	\forall	\exists	$(\forall x)(\underline{a}x) \vee (\forall y)(\underline{b}y)$	D
\exists	$(\exists x)(\underline{a}x) \Rightarrow (\exists y)(\underline{b}y)$	\exists	\vee	\vee	$(\forall x)(\underline{a}x) \vee (\forall y)(\underline{b}y)$	F

Lecture 4 : Details of QL-1 and QL-2 algebra

DEFINITION OF RELATIVE TRUTH VALUES IN SNS

By relative truth value we mean the truth value of a term or a relation relative to that of another. Thus in SNS $\underline{t}(\underline{b} | \underline{a})$ — "truth value of \underline{b} relative to \underline{a} " or "relative truth value of \underline{b} for \underline{a} " — is defined as the SNS truth value of $\underline{c} = s(kc)$ of the equivalence relation $\underline{b} \underline{E} \underline{a} = \underline{c}$. Then, we have

$$t_{\alpha} = \langle b | E | a \rangle = \langle b | a \rangle = t(b | a) \quad \text{say} \quad (1a)$$

$$t_{\beta} = \langle b | E^c | a \rangle = \langle b | E | a^c \rangle = \langle b | a^c \rangle = t(b | a^c) \quad \text{say} \quad (1b)$$

where the scalar product $\langle p | q \rangle$ of the two SNS vectors \underline{p} and \underline{q} is given by

$$\langle p | q \rangle = (p_{\alpha} \otimes q_{\alpha}^c) \quad (2)$$

Hence the relative truth value of \underline{b} for \underline{a} is

$$\underline{t}(\underline{b} | \underline{a}) = (t(b | a) t(b | a^c)) = (t_{\alpha} t_{\beta}) = \underline{t} \quad \text{say} \quad (3)$$

For $\underline{a} = T, \underline{b} = T$, and $\underline{a} = F, \underline{b} = F, \underline{t} = T$

$\underline{a} = T, \underline{b} = F$ or $\underline{a} = F, \underline{b} = T, \underline{t} = F$

and, for

one of $\underline{a}, \underline{b} = D, \underline{t} = D.$

The truth value of a binary relation for SNS input terms as well as the output term of an SNS unary relation can both be expressed in terms of truth values as given in the next sheet.

TRUTH VALUE OF UNARY AND BINARY RELATIONS IN SNS
VIA RELATIVE TRUTH VALUES

Binary relation

The Dirac product formula for the truth value of a binary relation $\underline{a} \underline{Z}(k, \ell) \underline{b} = \underline{c}$ can be put in terms of the relative truth values since $|\underline{A}(k, \ell)|$ and $|\underline{O}(k, \ell)|$ are given by the Boolean direct product and direct sum respectively, involving the generator states $s(1), s(2)$ of SNS, as

$$|\underline{A}(k, \ell)| = |s(k)\rangle \otimes \langle s(\ell)| ; |\underline{O}(k, \ell)| = |s(k)\rangle \oplus \langle s(\ell)| \quad (1)$$

Hence, for (c_α, c_β) of the relation $\underline{a} \underline{A}(k, \ell) \underline{b} = \underline{c}$, for inputs $s(ka)$ and $s(\ell b)$, we obtain

$$\begin{aligned} c_\alpha &= \langle s(ka) | \underline{A}(k, \ell) | s(\ell b) \rangle \equiv \langle s(ka) | s(k) \rangle \otimes \langle s(\ell) | s(\ell b) \rangle \\ &= t(s(ka) | s(k)) \otimes t(s(\ell) | s(\ell b)) = t(s(ka) | s(k)) \otimes t(s(\ell b) | s(\ell)) \end{aligned} \quad (2a)$$

$$\begin{aligned} c_\beta &= \langle s(ka) | \underline{O}(k, \ell) | s(\ell b) \rangle \equiv \langle s(ka) | s(k) \rangle \oplus \langle s(\ell) | s(\ell b) \rangle \\ &= t(s(ka) | s(k)) \oplus t(s(\ell) | s(\ell b)) = t(s(ka) | s(k)) \oplus t(s(\ell b) | s(\ell)) \end{aligned} \quad (2b)$$

Similarly, for the relation $\underline{a} \underline{O}(k, \ell) \underline{b} = \underline{c}$, it follows that

$$\begin{aligned} c_\alpha &= t(s(ka) | s(k)) \oplus t(s(\ell b) | s(\ell)) ; \\ c_\beta &= t(s(ka) | s(k)) \otimes t(s(\ell b) | s(\ell)) ; \quad \underline{c} = (c_\alpha, c_\beta) \end{aligned} \quad (3)$$

For $\underline{I}(k, \ell)$ and $\underline{E}(k, \ell)$, we use the standard formulae

$$\underline{I}(k, \ell) = \underline{O}(k^c, \ell) \quad \text{and} \quad \underline{E}(k, \ell) = \underline{A}(k, \ell) \oplus \underline{A}(k^c, \ell^c) \quad (4)$$

and work out the consequences.

Unary relation

The matrix product formula $\underline{a} \underline{Z}(k, \ell) = \underline{b}$ can also be written in terms of relative truth values. For the input $s(ka)$, the output $s(\ell b)$, for $\underline{A}(k, \ell)$ and $\underline{O}(k, \ell)$, are as follows.

$$\langle s(\ell b) | = \langle s(ka) | \underline{A}(k, \ell) | = \langle s(ka) | s(k) \rangle \otimes \langle s(\ell) | = t(s(ka) | s(k)) \otimes \langle s(\ell) | \quad (5a)$$

$$\langle s(\ell b) | = \langle s(ka) | \underline{O}(k, \ell) | = \langle s(ka) | s(k) \rangle \oplus \langle s(\ell) | = t(s(ka) | s(k)) \oplus \langle s(\ell) | \quad (5b)$$

These formulae are much faster for computer implementation than the vector-matrix multiplication formulae which were developed in Lecture 1.

IMPLEMENTATION OF SNS MATRIX RELATIONS VIA RELATIVE TRUTH VALUES

It will be noticed that only three types of BA-2 operations (1), (2), (3) are needed for matrix relations and Boolean relations in SNS, in terms of the BA-1 operators \oplus , \otimes and \circ .

$$\langle a | b \rangle = (a_\alpha \otimes b_\alpha) \oplus (a_\beta \otimes b_\beta) = c \quad (1)$$

$$a \oplus \langle b | = (a \oplus b_\alpha \quad a \oplus b_\beta) ; \quad a \otimes \langle b | = (a \otimes b_\alpha \quad a \otimes b_\beta) \quad (2)$$

$$\langle a | \oplus \langle b | = (a_\alpha \oplus b_\alpha \quad a_\beta \oplus b_\beta) ; \quad \langle a | \otimes \langle b | = (a_\alpha \otimes b_\alpha \quad a_\beta \otimes b_\beta) \quad (3)$$

As will be seen later, for QL-2 matrix and Boolean operators, the same three types of vector operations are all that are needed, but with three-component vectors of the types $(a_\gamma \quad a_\delta \quad a_\epsilon)$ in BA-3. These Boolean operations will be very useful for programs at the machine language level in software and hardware in computers. For QL-1, "lattice"-type operators. Max (i, j) and Min (i, j), where $i, j = 1$ to 8, will also be needed. These will be explained later.

Examples of implementation of BA-2 relations

Example 1 : Binary relation

$$(\neg a \wedge b \wedge (a \vee \neg b) \supset c) \equiv s(2a), s(1b), \underline{a} \underline{0}(1,2) \underline{b} = \underline{c} \mapsto s(kc)$$

$$\langle s(2a) | s(1) \rangle \oplus \langle s(1b) | s(2) \rangle = 0 \oplus 0 = 0 = c_\alpha ;$$

$$\langle s(2a) | s(2) \rangle \oplus \langle s(1b) | s(1) \rangle = 1 \oplus 1 = 1 = c_\beta ; \quad s(kc) = (0 \ 1) \equiv \underline{c} = F$$

Example 2 : Unary relation — implication

$$a \wedge (a \Rightarrow b) \supset b \equiv s(1a), \underline{a} \underline{1}(1,1)(\underline{0}(2,1)) = \underline{b} \mapsto s(\ell b)$$

$$\equiv \langle s(1a) | s(2) \rangle \oplus \langle s(1) | = 0 \oplus (1 \ 0) = (0 \oplus 1 \quad 0 \oplus 0) = (1 \ 0) \\ = s(\ell b) = \underline{b} = T$$

Example 3 : Unary relation — contradiction from conjunction

$$a \wedge (\neg a \wedge b) \supset X \equiv s(1a), \underline{a} \underline{A}(2,1) = \underline{b} \mapsto s(\ell b)$$

$$\equiv \langle s(1a) | s(2) \rangle \otimes \langle s(1) | = 0 \otimes (1 \ 0) = (0 \ 0) = s(\ell b) = \underline{b} = X$$

(Note that the contradiction between the input a , and $\neg a$ in the conjunction relation $(\neg a \wedge b)$, is shown up by the contradictory state given for the output by the implementation of the relation with the contradictory input.)

RELATIVE TRUTH VALUES IN QUANTIFIER LOGIC

In SNS, since the Boolean vectors have only two components (α, β) , the relative truth value is symmetric in \underline{a} and \underline{b} and $\underline{t}(\underline{a} | \underline{b}) = \underline{t}(\underline{b} | \underline{a})$. This is, however, not the case in QL. Here the truth value of \underline{b} relative to \underline{a} is defined in a manner analogous to (1a) and (1b) as follows.

$$\underline{t}(\underline{b} | \underline{a}) = (t_\alpha, t_\beta) \text{ where } t_\alpha = \langle \underline{b} | \underline{a} \rangle, \quad t_\beta = \langle \underline{b} | \underline{a}^c \rangle \quad (1)$$

The consequence is that the Boolean 3-vectors for \underline{a} and \underline{b} have the following inclusion properties for $\underline{t}(\underline{b} | \underline{a}) = T, F, D$.

$$\underline{b} \cap \underline{a}^c = \emptyset \mapsto \underline{t}(\underline{b} | \underline{a}) = T; \quad \underline{b} \cap \underline{a} = \emptyset \mapsto \underline{t}(\underline{b} | \underline{a}) = F \quad (2 \text{ a, b})$$

$$\text{Both } \underline{b} \cap \underline{a}^c \text{ and } \underline{b} \cap \underline{a} \neq \emptyset \mapsto \underline{t}(\underline{b} | \underline{a}) = D \quad (2 \text{ c})$$

Thus if the quantifier state of \underline{b} is completely contained in that of \underline{a} , then the relative truth value of \underline{b} for \underline{a} is T, and, similarly, if the QBA vector for \underline{b} is completely contained in that of \underline{a}^c , then the relative truth value is F. If the QBA vector \underline{b} is not completely contained in either \underline{a} or \underline{a}^c , then the relative truth value is D.

Examples

$$\begin{array}{ll} \underline{t}(\forall | \forall) = T; & \underline{t}(\forall | \vee) = F; \quad \underline{t}(\forall | \emptyset) = F; \quad \underline{t}(\forall | \exists) = T \\ \underline{t}(\exists | \forall) = D; & \underline{t}(\exists | \exists) = T; \quad \underline{t}(\exists | \vee) = D; \quad \underline{t}(\exists | \emptyset) = F \\ \underline{t}(\emptyset | \forall) = F; & \underline{t}(\emptyset | \vee) = T; \quad \underline{t}(\emptyset | \exists) = F; \quad \underline{t}(\emptyset | \emptyset) = T \\ \underline{t}(\forall | \exists) = F; & \underline{t}(\exists | \forall) = F; \quad \underline{t}(\exists | \exists) = D; \quad \underline{t}(\exists | \emptyset) = T \end{array}$$

QL-2 ALGEBRA FOR MATRIX CONNECTIVES1. GENERAL MATRIX RELATION

A general QL-2 matrix relation has the form (1), with the operator \underline{Z} being represented by a 3×3 Boolean matrix. The theory of this is given in Curr. Sci. Part II 52 (1983) 335.

$$\underline{ax} \underline{Z} \underline{by} : \underline{ax}, \underline{by} = q(1) \text{ to } q(8), |\underline{Z}| = |Z_{\lambda\mu}|, \lambda, \mu = \gamma, \delta, \epsilon \quad (1)$$

Definition of $Z_{\lambda\mu}$: Following the definition of a general matrix relation in BA-n (Curr. Sci. Part I 52 (1983) 292), the components of the matrix $|Z_{\lambda\mu}|$ can be defined in terms of the outputs $\underline{b}(1), \underline{b}(3), \underline{b}(5)$ corresponding to the inputs $\underline{a} = q(1), q(3), q(5)$ for the relation \underline{Z} , as in (2)

$$\begin{aligned} \underline{a} = q(1) &= (1 \ 0 \ 0) \implies \underline{b}(1) = (Z_{\gamma\gamma} \ Z_{\gamma\delta} \ Z_{\gamma\epsilon}) \\ \underline{a} = q(3) &= (0 \ 1 \ 0) \implies \underline{b}(3) = (Z_{\delta\gamma} \ Z_{\delta\delta} \ Z_{\delta\epsilon}) \\ \underline{a} = q(5) &= (0 \ 0 \ 1) \implies \underline{b}(5) = (Z_{\epsilon\gamma} \ Z_{\epsilon\delta} \ Z_{\epsilon\epsilon}) \end{aligned} \quad (2)$$

For such general relations, which need not be restricted to logical relations, the matrix $|\underline{Z}|$ can be any one of the $2^9 (= 512)$ possible 3×3 Boolean matrices.

Unary and binary relations

Unary relation: $q(iax), \underline{ax} \underline{Z} = \underline{by} \mapsto q(jby)$ is implemented as

$$\langle q(iax) | \underline{Z} | = \langle q(jby) | \quad (3a)$$

using the BVMF formula $(\langle a | \underline{Z} | = \langle b |) \equiv \bigoplus_{\lambda} a_{\lambda} \otimes Z_{\lambda\mu} = b_{\mu}; \lambda, \mu = \gamma, \delta, \epsilon$ (3b)

Binary relation: $q(iax), q(jby) \underline{ax} \underline{Z} \underline{by} = \underline{c}$ is implemented as

$$\langle q(iax) | \underline{Z} | q(jby) \rangle = c_{\alpha}; \langle q(iax) | \underline{Z}^c | q(jby) \rangle = c_{\beta}; \underline{c} = (c_{\alpha} \ c_{\beta}) \quad (4a)$$

and the Dirac product has the BVMF expansion

$$\langle a | \underline{Z} | b \rangle = \bigoplus_{\lambda} \bigoplus_{\mu} a_{\lambda} \otimes Z_{\lambda\mu} \otimes b_{\mu}; \lambda, \mu = \gamma, \delta, \epsilon \quad (4b)$$

QL-2 ALGEBRA FOR MATRIX CONNECTIVES2. LOGICAL RELATIONS VIA QL-2A IMPLEMENTATION

For logical relations in QL-2, we do not need all the 512 matrices, but only the logical matrix operators associated with the relations in (1)

$$\underline{ax} \left[q(iZx) \underline{Z}(k, \ell) q(jZy) \right] \underline{by}, \quad \underline{Z} = \underline{A}, \underline{Q}(\underline{I}), \underline{E}\text{-type} \quad (1)$$

The operational matrix of the connective for the relation within the square brackets in (1) is the one that is applicable. Thus, the matrix is represented by the symbol $\underline{Z}(i_Z, j_Z; k, \ell)$ and is equivalent to the operator $\underline{Z}(i, j)$ given by (2)

$$\underline{Z}(i_Z, j_Z; k, \ell) = \underline{Z}(i, j); \text{ where } i = i_Z, i_Z^c \text{ for } k = 1, 2; \\ j = j_Z, j_Z^c \text{ for } \ell = 1, 2 \quad (2)$$

This follows from the notation adopted, namely that QL-2 relation of the type $\neg(\forall x)(ax) \underline{Z} \neg(\exists y)(by)$ is denoted by $(\forall x)(ax) \underline{Z}(2,2) (\exists y)(by)$, in which the value of k or ℓ stand for the complementation of the corresponding QBA states associated with x and y respectively. Thereafter, the complemented quantifier (e.g. $\neg(\forall x)(ax)$) is converted to the QBA state $((\wedge x)(ax) \equiv q(2ax))$.

The implementation of a general QL-2A relation of the type $\underline{ax} \underline{Z}(i,j) \underline{by}$, with $\underline{Z}(i,j)$ restricted to the logical matrix operators $|\underline{A}(i,j)|, |\underline{Q}(i,j)|, |\underline{I}(i,j)|$ and $|\underline{E}(i,j)|$, each of which are 64 in number (according to $i, j = 1$ to 8), is best done, not by using the 3x3 Boolean matrices of the general QL-2 form, but via relative truth value calculations as given below. Thus the matrices for $|\underline{A}(i,j)|, |\underline{Q}(i,j)|, |\underline{I}(i,j)|$ and $|\underline{E}(i,j)|$ are definable in a manner closely similar to that for the 2x2 matrices of SNS as in (3a,b).

$$|\underline{A}(i,j)| = |q(1)\rangle \otimes \langle q(j)|; |\underline{Q}(i,j)| = |q(1)\rangle \oplus \langle q(j)| \quad (3a)$$

$$|\underline{I}(i,j)| = |\underline{Q}(i^c, j)|; |\underline{E}(i,j)| = |\underline{A}(i,j)| \oplus |\underline{A}(i^c, j^c)| \quad (3b)$$

These follow from the fact that the 3x3 truth table has 1's for $Z_{\lambda\mu}$ corresponding to $|q(1)\rangle \otimes \langle q(j)|$ and $|q(1)\rangle \oplus \langle q(j)|$ for the generator states $q(1), q(3), q(5)$, by definition, for the connectives \underline{A} and \underline{Q} .

QL-2 ALGEBRA FOR MATRIX CONNECTIVES3. IMPLEMENTATION OF QL-2A EQUATIONS VIA RELATIVE TRUTH VALUES

Thus, all the equations in sheet 2 for SNS can be taken over for QL-2A with $s(k)$, $s(\ell)$ being replaced by $q(ix)$ and $q(jy)$ respectively, and the implementation formulae for unary and binary relations formulated accordingly. These are as follows.

Logical relation : $\underline{ax} \underline{Z}(i,j) \underline{bx} , \underline{Z}(i,j) \equiv q(ix) \underline{Z}(1,1) q(jy) \quad (1)$

Binary relation

$q(iax), q(jby), \underline{ax} \underline{Z}(i,j) \underline{by} = \underline{c} \mapsto \underline{t}(\underline{Z} | q(iax), q(jby)) = s(kc) \quad (2a)$

$\underline{t}(q(iax) | q(ix)) = s(kx) ; \underline{t}(q(jby) | q(jy)) = s(\ell y) \quad (2b)$

$s_{\alpha}(kx) \otimes s_{\alpha}(\ell y) = s_{\alpha}(kc) ; s_{\beta}(kx) \oplus s_{\beta}(\ell y) = s_{\beta}(kc), \text{ for } \underline{Z} = \underline{A}(1,1) \quad (2c)$

$s_{\alpha}(kx) \oplus s_{\alpha}(\ell y) = s_{\alpha}(kc) ; s_{\beta}(kx) \otimes s_{\beta}(\ell y) = s_{\beta}(kc), \text{ for } \underline{Z} = \underline{O}(1,1) \quad (2d)$

yielding $s(kc) = (s_{\alpha}(kc) s_{\beta}(kc))$ and hence $\underline{t}(\underline{Z} | q(iax), q(jby))$.

For I-type and E-type, the implementation is done via suitable relations as in Eq.(3b) of the previous sheet.

Unary relation: $q(iax), \underline{ax} \underline{Z}(i,j) = \underline{bx} \mapsto q(jby) \quad (3a)$

We shall give the formulae for $\underline{Z} = \underline{A}$ and \underline{O} separately.

$\underline{Z} = \underline{A} : \langle q(iax) | q(ix) \rangle = a ; a \otimes \langle q(jy) | = \langle q(jby) | \quad (3b)$

$\underline{Z} = \underline{O} : \langle q(iax) | q(ix) \rangle = a ; a \oplus \langle q(jy) | = \langle q(jby) | \quad (3c)$

For $\underline{Z} = \underline{I}$ and \underline{E} , the corresponding formulae can be obtained in terms of the above equations for \underline{A} and \underline{O} -type matrix operators.

There is also a revision of the input, for $\underline{Z} = \underline{A}$, from $q(iax)$ to $q(ia_0x)$, using the vidya operator

$$q(ix) \underline{V} q(iax) = q(ia_0x) \quad (4)$$

The value of ϕ for $q(ia_0x)$ indicates contradiction between the input and the relation. Otherwise $q(ia_0x)$ gives the common information in $q(ix)$ and $q(iax)$.

Binary reverse relation : As in SNS, the unary relation

$\underline{a} \underline{Z}(i,j) = \underline{b}$ arises from the binary reverse relation as $\underline{ax} \underline{Z}(i,j) \underline{by} = \underline{c} = T \mapsto \underline{ax} \underline{Z}(i,j) = \underline{by}$, and $\underline{by} \underline{Z}^{tc}(i,j) = \underline{ax} \quad (5a)$

For $\underline{c} = F, D$ also, it takes similar forms

$\underline{ax} \underline{Z}(i,j) \underline{by} = \underline{c} = F \mapsto \underline{ax} \underline{Z}^c(i,j) = \underline{by}$, and $\underline{by} \underline{Z}^{tc}(i,j) = \underline{ax} \quad (5b)$

$\underline{ax} \underline{Z}(i,j) \underline{by} = \underline{c} = D \mapsto \underline{ax} \underline{D} = \underline{by}$ and $\underline{by} \underline{D} = \underline{ax}$, where $\underline{D} = \underline{O}(7,7) \quad (5c)$

PRACTICAL IMPLEMENTATION OF BINARY AND UNARY RELATIONS
IN QL-2A

Example 1 : Binary relation : Consider the inputs $(\forall x)(\underline{ax})$, $(\exists y)(\underline{by})$ in the relation

$$\neg (\exists x)(\underline{ax}) \vee (\forall y)(\neg \underline{by}) = \underline{c} \mapsto \underline{t}(\underline{z} \mid \underline{a}, \underline{b}) = F \quad (1)$$

The QL-2A formula for this is

$$q(1ax), q(6by), q(6^c x) \underline{Q}(1,1) q(1^n y) \mapsto s(kc) \quad (2)$$

Since $6^c = 5$ and $1^n = 5$, we recast Eq.(2) in the form

$$q(1ax), q(6by), \underline{ax} \underline{Q}(5,5) \underline{by} \mapsto s(kx) = \underline{t}(q(1ax) \mid q(5x)) ; \quad (3)$$

$$s(\ell y) = \underline{t}(q(6by) \mid q(5y))$$

where

$$s_\alpha(kx) = \langle q(1) \mid q(5) \rangle = \langle 1 \ 0 \ 0 \mid 0 \ 0 \ 1 \rangle = 0$$

$$s_\alpha(ky) = \langle q(6) \mid q(5) \rangle = \langle 1 \ 1 \ 0 \mid 0 \ 0 \ 1 \rangle = 0$$

then $s_\alpha(kx) \oplus s_\alpha(ky) = 0 = s_\alpha(kc)$ (4a)

$$s_\beta(kx) = \langle q(1) \mid q(6) \rangle = \langle 1 \ 0 \ 0 \mid 1 \ 1 \ 0 \rangle = 1$$

$$s_\beta(\ell y) = \langle q(6) \mid q(6) \rangle = \langle 1 \ 1 \ 0 \mid 1 \ 1 \ 0 \rangle = 1$$

$$s_\beta(kx) \otimes s_\beta(\ell y) = 1 = s_\beta(kc) \quad (4b)$$

so that $s(kc) = (0 \ 1) = F$ (5)

Example 2 : Unary relation of implication: Consider the equation

$$(\forall x)(\underline{ax}), (\exists x)(\underline{ax}) \implies (\exists y)(\underline{by}) \mapsto (\exists y)(\underline{by}) \quad (6)$$

The BVMF formula in QL-2A for this is

$$q(1ax), q(6^c x) \underline{Q}(2,1) \mapsto q(jby) = \langle q(1ax) \mid q(6^c x) \rangle \oplus \langle q(1y) \mid$$

$$= \langle \forall \mid \exists \rangle \oplus \langle q(6) \mid = 0 \oplus (1 \ 1 \ 0) = (1 \ 1 \ 0) = (\exists y)(by) \quad (7)$$

Example 3 : Unary relation involving conjunction

$$(\forall x)(\underline{ax}), (\exists x)(\neg \underline{ax}) \wedge (\forall y)(\underline{by}) \mapsto \text{contradiction} \quad (8)$$

The BVMF equation is

$$q(1ax), q(6^n x) \underline{A}(1,1) (\equiv q(2x) \underline{A}(1,1)) \mapsto \langle q(1ax) \mid q(2x) \rangle \oplus \langle q(1y) \mid$$

$$\langle \forall \mid \forall \rangle \oplus \langle \neg \mid \neg \rangle = 0 \otimes (1 \ 0 \ 0) = (0 \ 0 \ 0) = \phi, \text{ indicating contradiction} \quad (9)$$

It may also be checked via the vidya operator as below:

$$q(1ax) \underline{V} q(6^n x) = (1 \ 0 \ 0) \otimes (0 \ 1 \ 1) = (0 \ 0 \ 0) = \phi, \quad (10)$$

indicating the origin of the contradiction.

MATRIX RELATIONS IN QL-1 ALGEBRA - 1Logical relation

If $\underline{a}x, \underline{b}x$ are the quantifier states of the $(n,2)$ -vectors $\underline{a} \equiv (\underline{a}_1, \dots, \underline{a}_x, \dots, \underline{a}_n)$ and $\underline{b} \equiv (\underline{b}_1, \dots, \underline{b}_x, \dots, \underline{b}_n)$, then the general form of a QL-1 logical relation is

$$\underline{a}x \underline{Z}(k, \ell) \underline{b}x \equiv (\underline{a}x \underline{Z}(k, \ell) \underline{b}x) \text{ for each } x = 1 \text{ to } n, \\ \text{for } \underline{Z} \text{ of the types } \underline{A}, \underline{O(I)}, \underline{E} \text{ in BA-2 algebra of SNS} \quad (1)$$

From the definition, it follows that

$$\underline{a}'x \underline{Z}(k, \ell) \underline{b}x \equiv \underline{a}x \underline{Z} \underline{b}x \quad (2a)$$

where

$$\underline{a}x = \underline{a}'x \sigma(k), \text{ where } \sigma(k) = \underline{E}, \underline{N}, \text{ according as } k = 1, 2 \quad (2b)$$

and

$$\underline{b}x = \underline{b}'x \sigma(\ell), \text{ where } \sigma(\ell) = \underline{E}, \underline{N}, \text{ according as } \ell = 1, 2 \quad (2c)$$

Binary relation

For the inputs $q(\underline{ia}x)$ and $q(\underline{jbx})$, the r.h.s of (2a) can be shown to take the following forms

$$q(\underline{ia}x) \bigoplus_L q(\underline{jbx}) = q(\underline{icx}) \text{ for } \underline{Z} = \underline{O}; \bigoplus_L = \text{"lattice into"} \quad (3a)$$

and

$$q(\underline{ia}x) \star_L q(\underline{jbx}) = q(\underline{icx}) \text{ for } \underline{Z} = \underline{A}; \star_L = \text{"lattice star"} \quad (3b)$$

where the operators \bigoplus_L and \star_L are definable, in terms of the operators "lattice sum" (\bigoplus_L) and "lattice product" (\bigotimes_L), in a form analogous to the corresponding Boolean operators "into" (\bigotimes), "star" (\star), "sum" (\bigoplus), "product" (\bigotimes), considered for QL-1 Boolean operators in sheet 12 of Lecture 3.

MATRIX RELATIONS IN QL-1 ALGEBRA - 2Lattice operators

We take the generators of QBA, namely \vee, \wedge, \oplus to form a "lattice" with the inclusion properties $\vee > \wedge > \oplus$ (following from the fact that the range of \mathcal{V} in the definition of the quantifier has the inclusion property $R_{\vee}(\mathcal{V}) \supset R_{\wedge}(\mathcal{V}) \supset R_{\oplus}(\mathcal{V})$). Then, we apply the standard definitions of lattice sum and lattice product to combine these generator states to obtain the formulae

$$\underline{a} \underset{L}{\vee} \underline{b} = \underline{c}, \underline{c} = \text{Max}(\underline{a}, \underline{b}) ; \underline{a} \underset{L}{\wedge} \underline{b} = \underline{c}, \underline{c} = \text{Min}(\underline{a}, \underline{b}) \quad (4a, b)$$

However, it is useful to employ the indices $i = 1, 3, 5$ for $q(i) = \vee, \wedge, \oplus$ for computations, because of which we obtain the Eqs (5a, b), in order to satisfy Eqs (3a, b)

$$\begin{aligned} q(i) \underset{L}{\oplus} q(j) &= q(k), \quad k = \text{Min}(i, j) \\ q(i) \underset{L}{\wedge} q(j) &= q(k), \quad k = \text{Max}(i, j) \end{aligned} \quad (5a, b)$$

It is then possible to define the QL-1 "and"(\underline{A}) and "or"(\underline{O}) for the generators as follows.

$$\begin{aligned} q(i) \underline{O} q(j) &= q(i) \underset{L}{\oplus} q(j) = q(i) \underset{L}{\oplus} q(j) \text{ for all } (i, j), \\ \text{except } (3, 3), \text{ for which } q(3) \underset{L}{\oplus} q(3) &= q(3) \underset{L}{\oplus} q(1) = q(6) \\ (\underline{\wedge} \underset{L}{\oplus} \underline{\wedge} &= \underline{\exists}) \end{aligned} \quad (6a)$$

$$\begin{aligned} q(i) \underline{A} q(j) &= q(i) \underset{L}{\wedge} q(j) = q(i) \underset{L}{\wedge} q(j) \text{ for all } (i, j), \\ \text{except } (3, 3), \text{ for which } q(3) \underset{L}{\wedge} q(3) &= q(3) \underset{L}{\wedge} q(5) = q(2) \\ (\underline{\wedge} \underset{L}{\wedge} \underline{\wedge} &= \underline{\wedge}) \end{aligned} \quad (6b)$$

The exceptions arise for the same reason as that for the Boolean operators $\underset{L}{\oplus}$ and $\underset{L}{\wedge}$ in the QL-1 definition of these in terms of the Boolean sum and product for $q(3) \underset{L}{\oplus} q(3)$ and $q(3) \underset{L}{\wedge} q(3)$.

For a general QBA state which is the Boolean sum of one, two, or three, generator states, the Boolean sum representation of the QBA state is employed in the lattice sum and product formulae (6a) and (6b) with $\underline{0}$ and \underline{A} , and expanded as a Boolean sum of conjunctions and disjunctions between pairs of generator states for which (6a) and (6b) are used. Thus,

$$\begin{aligned} & (q(i_1) \oplus q(i_2)) \underline{A} (q(j_1) \oplus q(j_2)) \\ &= (q(i_1) \underline{A} q(j_1)) \oplus (q(i_2) \underline{A} q(j_1)) \oplus (q(i_1) \underline{A} q(j_2)) \\ & \quad \oplus (q(i_2) \underline{A} q(j_2)) \end{aligned} \quad (7)$$

and similar formulae for $\underline{0}$. These will give the results given in Table 1(a, b) for the two operators \underline{A} and $\underline{0}$.

of binary relations

Table 1. QL-1 truth tables/for the generators states of QBA

(a) AND (\underline{A})

\underline{A}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Phi}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$

(b) OR ($\underline{0}$)

$\underline{0}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	\underline{V}	\underline{V}
$\underline{\Sigma}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Sigma}$
$\underline{\Phi}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$

For the connectives \underline{I} and \underline{E} of QL-1 we use the relations

$$\underline{I}(i, j) = \underline{0}(i^n, j) \quad ; \quad \underline{E}(i, j) = \underline{A}(i, j) \oplus \underline{A}(i^n, j^n) \quad (8)$$

The truth tables for the generator states, in QL-1, for these operators $\underline{I}(1, 1)$ and $\underline{E}(1, 1)$ are also given in Tables 1(c, d).

(c) IMPLY (\underline{I})

\underline{I}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Sigma}$
$\underline{\Phi}$	\underline{V}	\underline{V}	\underline{V}

(d) EQUIVALENT (\underline{E})

\underline{E}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
\underline{V}	\underline{V}	$\underline{\Sigma}$	$\underline{\Phi}$
$\underline{\Sigma}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Sigma}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Sigma}$	\underline{V}

The full 8x8 tables for \underline{A} and $\underline{0}$ are given in the next sheet.

Table 2. Full multiplication table in QL-1 for the matrix connectives \underline{A} and \underline{Q}

(a) AND ($\underline{ax} \underline{A} \underline{bx} = \underline{cx}$)

$\underline{a} \quad \underline{b}$	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{V}	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{E}	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\vee}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\phi}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\phi}$
$\underline{\Sigma}$	$\underline{\Sigma}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\vee}$	$\underline{\phi}$
$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Theta}$	$\underline{\Theta}$	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

(b) OR ($\underline{ax} \underline{Q} \underline{bx} = \underline{cx}$)

$\underline{a} \quad \underline{b}$	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{V}	\underline{V}	\underline{V}	\underline{V}	\underline{V}	\underline{V}	\underline{V}	\underline{V}	$\underline{\phi}$
\underline{E}	\underline{V}	\underline{E}	\underline{E}	\underline{E}	\underline{E}	\underline{E}	\underline{E}	$\underline{\phi}$
$\underline{\vee}$	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Phi}$	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\Sigma}$	\underline{V}	\underline{E}	\underline{E}	$\underline{\Sigma}$	\underline{E}	\underline{E}	\underline{E}	$\underline{\phi}$
$\underline{\Delta}$	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Theta}$	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\Theta}$	\underline{E}	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

MATRIX RELATIONS IN QL-1 ALGEBRA - 5

PRACTICAL APPLICATION IN QL-1A EQUATIONS

Binary relations : The binary forward relation $\underline{ax} \underline{Z} \underline{bx} = \underline{cx}$ in QL-1 is used for calculating the truth value of the standard predicate formula (1) (QL-1A) for the inputs given therein.

$$(q_z x)(\underline{ax} \underline{Z}(k, \ell) \underline{bx}) = \underline{t}(\underline{Z} | \underline{ax}, \underline{bx}), \text{ for inputs } \underline{ax} = q(1ax), \underline{bx} = q(1bx) \quad (9)$$

This is done by first implementing the corresponding QL-1 binary relation

$$\underline{ax} \underline{Z} \underline{bx} = \underline{cx}, \text{ for inputs } q(1^{\sigma(k)} ax), q(1^{\sigma(\ell)} bx) \mapsto q(1cx) \quad (10a)$$

$$\text{where } \sigma(k) = e, n, \text{ for } k = 1, 2; \sigma(\ell) = e, n \text{ for } \ell = 1, 2 \quad (10b)$$

Then, the relative truth value of $q(1cx)$ for $q(1Zx)$ yields $\underline{t}(\underline{Z} | \underline{a}, \underline{b})$, as in (11).

$$\underline{t}(\underline{Z} | q(1ax), q(1bx)) = \underline{t}(q(1cx) | q(1Zx)) \quad (11)$$

Then the algorithm composed of (5, 6, 7, 8) (or Tables 2(a, b)) will yield $q(1cx)$ from (10a), and hence the truth/ ^{value} of (9) via (11).

Example: Suppose, we have the inputs for the relation in (12):

$$(\forall x)(\underline{ax}) (\equiv q(1ax)), (\exists x)(\underline{bx}) (\equiv q(6bx)), (\forall x)(\underline{ax} \wedge \neg \underline{bx}) \quad (12)$$

The BVMF formula for (12) is

$$q(1ax), q(6bx), q(1Zx) (\underline{ax} \underline{A}(1, 2) \underline{bx}) \quad (13a)$$

which leads to the QL-1 binary relation

$$\begin{aligned} q(1ax), q(2bx), \underline{ax} \underline{A} \underline{bx} &\equiv (q(1) \underline{A} q(5)) \oplus (q(1) \underline{A} q(3)) \\ &= q(5) \oplus q(3) = q(2) = q(1cx) \end{aligned} \quad (14a)$$

$$\text{Hence, } \underline{t}(\underline{Z} | \underline{a}, \underline{b}) = \underline{t}(q(1cx) | q(1Zx)) = \underline{t}(\underline{A} | \underline{V}) = (0 \ 1) = F \quad (14b)$$

Thus, the relation in (12) is negated for the inputs shown therein.

MATRIX RELATIONS IN QL-1 ALGEBRA-6

PRACTICAL APPLICATION

UNARY QL-1A RELATIONS IN PREDICATE CALCULUS

The unary relation in predicate calculus having the QL-1A form

$$q(iax), q(izx) (\underline{ax} \underline{z}(k, \ell) = \underline{bx}) \mapsto q(jbx) \quad (15a)$$

is equivalent to a binary reverse relation in QL-1 for the quantifier ($\underline{q_z}x$) as output. Thus the forward binary QL-1A relation equivalent to (15a) is

$$q(iax) \underline{z}(k, \ell) q(jbx) = q(izx) \text{ or } \underline{ax} \underline{z} \underline{bx} = \underline{cx} = (\underline{q_z}x) \quad (15b,c)$$

which can be put in the form of the binary reverse relation in QL-1, namely

$$\underline{ax}(\underline{q_z}x, \underline{z}) = \underline{bx} \quad (16)$$

Consequently, the truth tables for (15a) can be worked out for the generator states \vee, \sum, \oplus , using Tables 1(a-d).

The results so obtained are shown in Tables 3(a-d) in sheet number 15 (for details, see MR-54, pages 32-53). An empirical formula for representing the data in Table 3 is also given in Eqs. (17) and (18) in that sheet. For ready reference, the four 8x8 truth tables for the data in Table 3 are appended as Tables 4(a-d) in sheet numbers 16, 17.

PRACTICAL APPLICATION

UNARY RELATION IN PREDICATE CALCULUS

Table 3. Truth tables for the QL-1 binary reverse relations for the generator states of QBA

(a) $\underline{a}(\underline{c}, \underline{A}) = \underline{b}$

\underline{a}	\underline{c}	\forall	\exists	Φ
\forall	\forall	\exists	Φ	
\exists	Φ	\exists	\wedge	
Φ	Φ	Φ	Δ	

(b) $\underline{a}(\underline{c}, \underline{Q}) = \underline{b}$

\underline{a}	\underline{c}	\forall	\exists	Φ
\forall	Δ	Φ	Φ	
\exists	\exists	\wedge	Φ	
Φ	\forall	\exists	Φ	

(c) $\underline{a}(\underline{c}, \underline{I}) = \underline{b}$

\underline{a}	\underline{c}	\forall	\exists	Φ
\forall	\forall	\exists	Φ	
\exists	\exists	\wedge	Φ	
Φ	Δ	Φ	Φ	

(d) $\underline{a}(\underline{c}, \underline{E}) = \underline{b} = \underline{b}(\equiv (\underline{a} \underline{E} \underline{c} = \underline{b}))$

\underline{a}	\underline{c}	\forall	\exists	Φ
\forall	\forall	\exists	Φ	
\exists	\exists	Δ	\exists	
Φ	Φ	\exists	\forall	

The following empirical formulae can be used to obtain the output $q(jb)$ for the relation (17) :

$$q(ia) (q(kc), \underline{Z}) = q(jb) \quad (17)$$

$$\underline{A} : k < i, q(8) = q(j) ; k \geq i, \bigoplus_{\ell=0}^L q(k - i + 1 + 2\ell) = q(j), \quad L = (i - 1)/2 \quad (18a)$$

$$\underline{Q} : k > i, q(8) = q(j) ; k \leq i, \bigoplus_{\ell=0}^L q(k + 2\ell) = q(j), \quad L = (7 - i)/2 \quad (18b)$$

$$\underline{E} : q(i) (q(k), \underline{E}) \equiv q(1) \underline{E} q(k) = q(j) \quad (18c)$$

$$\underline{I} : q(i) (q(k), \underline{I}) \equiv q(6 - i) (q(k), \underline{Q}) = q(j) \quad (18d)$$

Table 4. Full 8x8 tables for QL-1 unary matrix connectives

(a) $\underline{ax} (\underline{cx}, \underline{A}) = \underline{bx}$

$\underline{a} \backslash \underline{c}$	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{V}	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{E}	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\vee}$	$\underline{\phi}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Sigma}$	$\underline{\phi}$	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\phi}$
$\underline{\Delta}$	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Theta}$	\underline{V}	\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

(b) $\underline{ax} (\underline{cx}, \underline{O}) = \underline{bx}$

	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\phi}$
\underline{V}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\Delta}$	$\underline{\Delta}$
\underline{E}	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Delta}$
$\underline{\vee}$	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Delta}$
$\underline{\Phi}$	\underline{V}	\underline{E}	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	
$\underline{\Sigma}$	\underline{E}	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\phi}$	$\underline{\vee}$	$\underline{\Delta}$	\underline{E}
$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\vee}$	$\underline{\Delta}$	$\underline{\Delta}$
$\underline{\Theta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\vee}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Delta}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

Table-4. Full 8x8 tables for QL-1 unary matrix connectives

(c) $\underline{ax} (\underline{cx}, \underline{I}) = \underline{bx}$

$\underline{a} \backslash \underline{c}$	\underline{V}	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{V}	\underline{V}	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\exists}$	$\underline{\exists}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\phi}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Phi}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Sigma}$	$\underline{\exists}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\phi}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\exists}$	$\underline{\phi}$
$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Theta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

(d) $\underline{ax} (\underline{cx}, \underline{E}) = \underline{bx}$

$\underline{a} \backslash \underline{c}$	\underline{V}	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
\underline{V}	\underline{V}	$\underline{\exists}$	$\underline{\wedge}$	$\underline{\Phi}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\exists}$	$\underline{\exists}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\wedge}$	$\underline{\wedge}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\exists}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Phi}$	$\underline{\Phi}$	$\underline{\wedge}$	$\underline{\exists}$	\underline{V}	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\Sigma}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Sigma}$	$\underline{\phi}$
$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\phi}$
$\underline{\Theta}$	$\underline{\Theta}$	$\underline{\Delta}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\Sigma}$	$\underline{\Delta}$	$\underline{\Theta}$	$\underline{\phi}$
$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$	$\underline{\phi}$

UNARY RELATION IN PREDICATE CALCULUS VIA QL-1 ALGEBRA

The unary relation (15a) in predicate calculus (QL-1A) is implemented in the form of the QL-1 binary reverse relation (19).

$$\underline{a}x (\underline{q}_Z x, \underline{Z}) = \underline{b}x ; \quad \underline{Z} = \underline{Z}(1, 1), \underline{Z} = \underline{A}, \underline{O}(\underline{I}), \underline{E} \quad (19)$$

Then,

$$\text{L.H.S of (15a)} \equiv (q(\underline{ia}'x), \underline{a}'x (q(\underline{iz}x), \underline{Z}) = \underline{b}'x) \mapsto q(\underline{jb}'x) \quad (20a)$$

where

$$q(\underline{ia}'x) = q^{\sigma(k)}(\underline{ia}x), \quad \sigma(k) = e, n, \quad \text{for } k = 1, 2 \quad (20b)$$

$$q(\underline{jb}'x) = q^{\sigma(\ell)}(\underline{jb}x), \quad \sigma(\ell) = e, n, \quad \text{for } \ell = 1, 2 \quad (20c)$$

With the sequence of the implementation as in (21),

$$q(\underline{ia}x) \xrightarrow{(20b)} q(\underline{ia}'x) \xrightarrow{(20a)} q(\underline{jb}'x) \xrightarrow{(20c)} q(\underline{jb}x) \quad (21)$$

we can use either the empirical formulae in (17) and (18) for the QL-1 binary reverse relation, or use the 8x8 tables, for obtaining the output $q(\underline{jb}x)$.

The revised input $q(\underline{ia}_0x)$ is obtained by means of the vidya check

$$q(\underline{ia}'x) \vee q(\underline{iz}x) = q(\underline{ia}_0'x) \quad (22)$$

If $q(\underline{ia}_0'x) = \phi$, it confirms the contradiction which would have been indicated by the null set for $q(\underline{jb}'x)$. If not, and $q(\underline{ia}_0'x) \neq q(\underline{ia}'x)$, it is the net resultant of the information contained in both the input and the relation, when the revised input \underline{a}_0x is

$$q(\underline{ia}_0'x) = q^{\sigma(k)}(\underline{ia}'x) \quad (23)$$

This feature, of a possible revision of the input, is common to all unary relations in QL, and requires a revision of Lecture-2, in the algorithm/for implementing a logical graph, more generally in QPL.

Generalization via the theory of relations in BVNF of the
clausal form of logic and its applications

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Generalization via the theory of relations in BVMF of the
clausal form of logic and its applications

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

CONTENTS

	Page No.
1. Introduction	1
2. BVMF formulae from the theory of relations	4
(a) General theory	4
(b) Unary implementation	8
(i) Description in terms of singular Boolean vectors	8
(ii) Equivalence of the BVMF Eq.(20) with the logical Eq.(21)	10
(iii) BVMF equations corresponding to the clausal relation in Eq.(3)	14
(c) Binary form of implementation of the standard clausal relation	16
3. Summary of BVMF formulae for the implementation of the four types of relations	17
(a) Type-1 relation	17
(b) Relations of the types 2, 3 and 4	22
4. BVMF formalism for a very general type of relation between logical polynomials	28
(a) General principles	28
(b) BVMF formulae for the unary and binary implementation of a Type-5 relation	32
(c) Proof of Eqs. (37) and (38)	36
(i) Steps involving unary for implementation	38
(ii) Second part of calculating the binary truth value	39
5. An interesting relation to quantifier states of the input and output of relations of Types 1 to 4.	41
(a) Significance of the binary truth value t_n	43
(b) Determination of quantifier states from truth values of relations	45

Generalization via the theory of relations in BVMF of the
clausal form of logic and its applications

1. Introduction

The BVMF theory of relations was presented in two sets of papers in Curr. Sci. (1983) and (1986) [1-4]. In the present report, we shall discuss the application of the EVM formalism for generalizing the clausal form of a relation R which is commonly taken to have the form of Eq.(1) (Kowalski [5]).

$$A_1 \wedge A_2 \wedge \dots \wedge A_K \stackrel{R}{=} B_1 \vee B_2 \vee \dots \vee B_L \quad (1)$$

As is well-known, this Eq.(1) can be represented as the disjunction of a number L of Horn clauses, each of which has the simpler form of Eq.(2).

$$A_1 \wedge A_2 \wedge \dots \wedge A_K \stackrel{R}{\Rightarrow} B_\ell, \quad \ell = 1 \text{ to } L \quad (2)$$

As commonly applied, in the clausal form, the disjunction of the inputs A_1 to A_K , or the conjunction of the outputs B_1 to B_L , have to be treated separately by writing them in terms of independent relations of the type of Eq.(1) in general. However, we find that it is possible to represent relations involving both conjunctions and disjunctions of the inputs

and the outputs in a uniform manner by using BVM formalism, and, at the same time, generalizing them so as to be applicable to the relation R between the members of two "universal" sets,

$\mathcal{A} \equiv \{A_\lambda\}$, $\lambda = 1$ to M , and $\mathcal{B} \equiv \{B_\mu\}$, $\mu = 1$ to N , of which the sets A_1 to A_K and B_1 to B_L of Eqs (1) and (2) are subsets $A \equiv \{A_{\lambda_k}\}$, $k = 1$ to K and $B \equiv \{B_{\mu_\ell}\}$, $\ell = 1$ to L in particular. Thus, in addition to Eq.(1), it is possible to give formulae to stand for relations of the type given in Eqs (3), (4) and (5) below

$$A_1 \vee A_2 \vee \dots \vee A_K \xRightarrow{R} B_1 \vee B_2 \vee \dots \vee B_L \quad (3)$$

$$A_1 \wedge A_2 \wedge \dots \wedge A_K \xRightarrow{R} B_1 \wedge B_2 \wedge \dots \wedge B_L \quad (4)$$

$$A_1 \vee A_2 \vee \dots \vee A_K \xRightarrow{R} B_1 \wedge B_2 \wedge \dots \wedge B_L \quad (5)$$

The theory presented here is based on an elementary relation of the type of Eq.(6) discussed in [1] , [3] , for the larger sets \mathcal{A} and \mathcal{B} , of the form

$$A_\lambda \quad R_{\lambda\mu} = B_\mu, \quad \lambda = 1 \text{ to } M, \quad \mu = 1 \text{ to } N \quad (6)$$

in which $R_{\lambda\mu} = 1$ if A_λ is related to B_μ , and 0 otherwise.

Using this, it is possible to give a basic relation which is

somewhat similar to the Horn clause, but which has a complementary form, namely

$$A_{\lambda_k} \xRightarrow{R} B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_L} \dots \vee B_{\mu_L}, k = 1 \text{ to } K, \\ \lambda_k \in 1 \text{ to } M, \mu_l \in 1 \text{ to } N \quad (7)$$

Equations of the type (7) can then be combined suitably so as to get the formulae for the standard clausal form of the relation, namely Eq.(1), as well as those having the form of Eqs. (3), (4) and (5). Eq.(7) has a single input and multiple outputs, since the BVM formalism uses the input to output procedure of implementing logical formulae, unlike the converse in the clausal form of implementation, so that the emphasis is on each of the inputs and their consequences.

Thus, general Boolean algebraic formulae representing the relation in Eq.(6) can be used to implement Eqs. (1), (3) (4) and (5) when we are given the elements $R_{\lambda\mu}$ of the relation, which is much more general than the formulae covered by the clausal form of a relation. However, if the information regarding $R_{\lambda\mu}$ is itself given via a clausal relation of the standard type as in Eq.(1), it is possible to

relational matrix so as to cover this contingency. In all cases, it is possible to obtain Boolean algebraic formulae, using BA-1 formalism as in classical 2-valued logic, to represent conjunctions and disjunctions, and Boolean M-vectors, N-vectors and $M \times N$ matrices to represent the subset A, subset B and the relational matrix $R_{\lambda\mu}$.

In what follows, we shall mainly adopt the treatment in [1] and [3], but develop a new notation so as to be consistent with the general theoretical treatments of propositional calculus and predicate calculus using quantifiers, as presented in Matphil Reports No. 52, 53, 54 ([6 - 8]). This is first considered in Section 2(a) and their application to Eq.(1) is considered in Sections 2(b) and (c). The extension to Eqs. (3), (4) and (5) is developed in Section 3. A still further extension to a general relation equation between logical polynomials $f_L(A_{\lambda_k})$ and $g_L(B_{\mu_L})$ is considered in Section 4, and an indication of how the four types of clausal relations can be connected to quantifier states is discussed in Section 5.

2. BVMF formulae from the theory of relations

(a) General theory

As mentioned above, the purpose of this section, is to introduce a notation which is consistent with that recently developed for propositional calculus and predicate calculus, but which is generalized to many-valued propositional calculus

which is the basis of the theory of relations. Suppose that we have two sets of atomic formulae, \mathcal{A} and \mathcal{B} , as in (8), with the relation (6), namely $A_\lambda R_{\lambda\mu} B_\mu$ between them :

$$\mathcal{A} \equiv \{A_\lambda\}, \quad \lambda = 1 \text{ to } M; \quad \mathcal{B} \equiv \{B_\mu\}, \quad \mu = 1 \text{ to } N \quad (8a,b)$$

The implementation of this relation can be performed in BVMF form as follows. If we consider the subset A of \mathcal{A} and the subset B of \mathcal{B} , consisting of the elements in (9a,b),

$$A \equiv (A_{\lambda_1}, A_{\lambda_2}, \dots, A_{\lambda_k}, \dots, A_{\lambda_K}), \\ \lambda_k (k = 1 \text{ to } K) \in 1 \text{ to } M \quad (9a)$$

$$B \equiv (B_{\mu_1}, B_{\mu_2}, \dots, B_{\mu_\ell}, \dots, B_{\mu_L}), \\ \mu_\ell (\ell = 1 \text{ to } L) \in 1 \text{ to } N \quad (9b)$$

then, the unary form of the relation, namely $A R = B$, has the following Boolean-algebraic representation and logical interpretation. The two Boolean vectors \underline{a} and \underline{b} representing the subsets A and B , have the components as in (10a) and (10b).

$$\underline{a} = (a_1, a_2, \dots, a_\lambda, \dots, a_M) \quad \text{with } a_\lambda = 1, \text{ for } \lambda = \lambda_k, \\ k = 1 \text{ to } K; \quad a_\lambda = 0, \text{ otherwise} \quad (10a)$$

$$\underline{b} = (b_1, b_2, \dots, b_\mu, \dots, b_N) \quad \text{with } b_\mu = 1, \text{ for } \mu = \mu_\ell, \\ \ell = 1 \text{ to } L; \quad b_\mu = 0, \text{ otherwise} \quad (10b)$$

Then, since the subsets A and B , defined by (9a,b) are the unions of K single-element subsets A_{λ_k} and L single-element

as the
 subsets B_{μ_ℓ} , the vectors \underline{a} and \underline{b} can be written/Boolean sums of
 "singular Boolean vectors" $\sigma(\lambda_k)$ and $\sigma(\mu_\ell)$, defined by
 Eqs. (11) and (12).

$$\sigma(\lambda_k) = (a_1, a_2, \dots, a_{\lambda_k}, \dots, a_M), \text{ with } a_{\lambda_k} = 1$$

$$\text{and } a_\lambda = 0 \text{ for } \lambda \neq \lambda_k \quad (11a)$$

$$\sigma(\mu_\ell) = (b_1, b_2, \dots, b_{\mu_\ell}, \dots, b_N), \text{ with } b_{\mu_\ell} = 1$$

$$\text{and } b_\mu = 0 \text{ for } \mu_\ell \neq \mu_\ell \quad (11b)$$

$$\underline{a} = \bigoplus_{k=1}^K \sigma(\lambda_k), \quad \underline{b} = \bigoplus_{\ell=1}^L \sigma(\mu_\ell) \quad (12a,b)$$

In order to indicate the parameters associated with the
 \underline{a} and \underline{b} , corresponding to the subsets A and B, we shall use
 the notation $v(a/\lambda K)$ and $v(b/\mu L)$ to represent the vectors
 \underline{a} and \underline{b} respectively in Eqs. (12a,b). This is closely similar
 to the notation involving the SNS 2-vector $s(ka)$ and the
 QL 3-vector $q(iax)$, employed ^{by us} for \underline{a} and \underline{a} in propositional
 calculus and predicate calculus, in [6], [7], [8]. Thus,

$$v(a \lambda K) = \bigoplus_{k=1}^K \sigma(\lambda_k) \longleftrightarrow A = \{A_{\lambda_k}\}, \quad k = 1 \text{ to } K \quad (13a)$$

$$v(b \mu L) = \bigoplus_{\ell=1}^L \sigma(\mu_\ell) \longleftrightarrow B = \{B_{\mu_\ell}\}, \quad \ell = 1 \text{ to } L \quad (13b)$$

Then, for a general form $a_\lambda R_{\lambda\mu} b_\mu$ of a relation as in Eq.(6), the BVMF representation $\underline{a} \underline{R} \underline{b}$ has, for its unary form, the matrix representation in the form (14),

$$\langle a | R | = \langle b | \quad (14)$$

in which $\langle a |$ and $\langle b |$ have the components given by (10), or by (11) and (12).

Its unary and binary forms of implementation are given by (15a,b)

Unary form of implementation

$$v(a \lambda K), \quad \underline{a} \underline{R} = \underline{b} \longmapsto v(b \mu L) \quad (15a)$$

Binary form of implementation

$$v(a \lambda K), v(b \mu L), \quad \underline{a} \underline{R} \underline{b} = c_\alpha \longmapsto t(\underline{a}, \underline{b} | \underline{R}) = c_\alpha \quad (15b)$$

where $\underline{a}, \underline{b}$ stand for general vectors as in (10a,b) representing the input and output of the relation, while $v(a \lambda K)$ and $v(b \mu L)$ (as in (13a,b)), represent the actual input and output vectors in a specific relation for which the general formula is implemented. We give below the actual formulae for the unary and binary forms of implementation as developed in [1] and [3], but modified to be in the notation adopted here.

(b) Unary implementation

We shall show that the BVM formula (14) is formally equivalent to the standard causal form of the relation as given in Eq.(1). We shall first give the algebraic formulae connected with the implementation of Eq.(15a) and then explain the significance of the equivalent logical relation.

(i) Description in terms of singular Boolean vectors

for $\underline{a} \underline{R} = \underline{b}$ in Eq. (15a)

The expansion of Eq.(14) leads to the row vectors $b_{\mu}(\lambda)$ of Eq.(16a)

$$b_{\mu}(\lambda) = a_{\lambda} R_{\lambda\mu}, \quad \mu = 1 \text{ to } N \quad (16a)$$

in which, by substituting $v(a \lambda K)$ from (13a), for \underline{a} in (15a), ^{we obtain} /

$$b_{\mu}(\lambda_k) = R_{\lambda_k\mu}, \text{ for } \lambda = \lambda_k, \text{ since } a_{\lambda_k} = 1 \text{ for } k = 1 \text{ to } K \quad (16b)$$

$$\text{and } b_{\mu}(\lambda) = 0, \text{ for } \lambda \neq \lambda_k \text{ since } a_{\lambda} = 0 \text{ for } \lambda \neq \lambda_k \quad (17)$$

Consequently, in the matrix product expansion of (14), namely

$$\sum_{\lambda=1}^M a_{\lambda} \omega R_{\lambda\mu} = b_{\mu}, \quad \mu = 1 \text{ to } N \quad (18)$$

the Boolean sum on the l.h.s need be carried out only for those

$\lambda = \lambda_k$, so that we obtain (19a)

$$\bigoplus_{k=1}^K (a_{\lambda_k} \otimes R_{\lambda_k \mu}) = \bigoplus_k R_{\lambda_k \mu} = \bigoplus_k b_{\mu}(\lambda_k) = b_{\mu},$$

$\mu = 1 \text{ to } N \quad (19a)$

The output vector b_{μ} can also be expressed in the form of Eq.(13b) as

$$\underline{b} = (b_1, \dots, b_{\mu}, \dots, b_N) = v(b_{\mu L}) = \bigoplus_{\ell=1}^L \sigma(\mu_{\ell}),$$

where $b_{\mu_{\ell}} = 1$ for $\mu = \mu_{\ell}$, and 0 for $\mu \neq \mu_{\ell}$ (19b)

Thus, by using the representation of $v(a_{\lambda K})$ and $v(b_{\mu L})$ in terms of the singular vectors $\sigma(\lambda_k)$ and $\sigma(\mu_{\ell})$ as in (13 a,b), Eqs. (19a,b) can be written in the form,

$$\bigoplus_{k=1}^K (\sigma(\lambda_k) \otimes R_{\lambda_k \mu}) = \bigoplus_{\ell=1}^L \sigma(\mu_{\ell}) \quad (20)$$

As we will show in the next subsection, this is logically equivalent to Eq.(1) which may be rewritten in the form

$$A \wedge A_{\lambda_2} \wedge \dots \wedge A_{\lambda_K} \xrightarrow{R} B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_N} \quad (21)$$

(ii) Equivalence of the BVMF Eq.(20) with the logical Eq.(21).

We shall illustrate this by an example and then generalize the result. Suppose that the set A of professors consists of four members A_1, A_2, A_3, A_4 and that the set B consists of four students B_1, B_2, B_3, B_4 who take courses under some one or more of the professors. The relational matrix $R_{\lambda\mu} (\lambda, \mu = 1 \text{ to } 4)$

$R_{\lambda\mu}$	B_1	B_2	B_3	B_4
A_1	1	0	1	0
A_2	1	1	0	1
A_3	0	0	1	1
A_4	0	1	0	1

(22a)

as given in (22a), indicates which professors give courses to which students. Thus, the first row (1 0 1 0) indicates that B_1 and B_3 take courses with professor A_1 ; the second row (1 1 0 1) indicates that the students B_1, B_2 and B_4 take courses with A_2 ; etc. Suppose we are required to find out which students take courses under A_1 and A_3 . Then the subset A of the professors under consideration will consist of (A_1, A_3) which has the Boolean-vector representation $\underline{a} = (1 \ 0 \ 1 \ 0)$. Hence Eq.(18) leads to the results as in (22b) below.

$$\begin{array}{rcl}
 (1 \ 0 \ 1 \ 0) & \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} & \begin{array}{l} (1 \ 0 \ 1 \ 0) \quad B_1 \text{ and } B_3 \\ \oplus (0 \ 0 \ 1 \ 1) \quad B_3 \text{ and } B_4 \\ \hline (1 \ 0 \ 1 \ 1) \quad B_1, B_3 \text{ and } B_4 \end{array}
 \end{array} \quad (22b)$$

This equation shows effectively the superposition of $b_{\mu}(\lambda_k)$ for $k = 1, 3$ leading to $b_{\mu} = (1 \ 0 \ 1 \ 1)$. On analyzing this as the Boolean sum on the r.h.s of Eq.(20), of $\sigma(\mu_{\ell})$, we notice that it is equivalent to the set of three students $\{B_1, B_3, B_4\}$, or $\mu_{\ell} = 1, 3$ or 4 . Hence the answer is that the students B_1, B_3, B_4 take courses under the professors A_1 and A_2 if we are given the relational matrix giving the distribution of professors and students as in (22b).

The main thing to be explained, in generalizing this to Eq.(21) as being the logical interpretation of Eq.(20), is the justification of the conjunction on the l.h.s of (21) for A_{λ_k} and the disjunction on the r.h.s for $B_{\mu_{\ell}}$. We wish to obtain the set of students associated with the given subset $A = \{A_{\lambda_1}, A_{\lambda_2}, \dots, A_{\lambda_K}\}$ of the full set of professors A_{λ} ($\lambda = 1$ to M). It will be seen that all these members A_{λ_k} ($k = 1$ to K) will be present (true) if $A_{\lambda_1} \wedge A_{\lambda_2} \wedge \dots \wedge A_{\lambda_K}$

is true. Consequently, the consequences of each set $\{B_{\mu}(\lambda_k)\}$ of the students associated with A_{λ_k} must be combined, so that when the resultant set of all the students who take courses under the ^{given subset} of the professors is evaluated, the students who belong to $B_{\mu}(\lambda_k)$ for each λ_k have to be included. The BVMF vector representing this is seen to be nothing but the multiple Boolean sum (\oplus) of $b_{\mu}(\lambda_k)$ as in (19a), which is the same as the l.h.s of (20). Note that effectively the conjunction of the members contained in the input subset $A = \{A_{\lambda_k}\}$ comes out as the multiple Boolean sum (or disjunction) of the corresponding Boolean vectors, namely $\bigoplus_{\lambda} b_{\mu}(\lambda_k)$ for the output vector $v(b_{\mu} L)$ in (19a) and in the l.h.s of Eq.(20). (As will be seen below in the next subsection (iii), if we have $A_{\lambda_1} \vee A_{\lambda_2} \vee \dots \vee A_{\lambda_K}$ in the l.h.s, as in the relation in Eq.(3), then the corresponding Boolean operation to the l.h.s of Eq.(20) would be the multiple Boolean product (\otimes)).

As regards the interpretation of the r.h.s of Eq.(20) as the disjunction of the elements $B_{\mu_1}, B_{\mu_2}, \dots, B_{\mu_L}$ in the r.h.s of (21), this question comes up only for a binary form

of implementation of the relation — namely to find out the truth value of the relation, given $v(a \lambda K)$ and $v(b \mu L)$, as in (15b). Here, we shall only mention that the effective Boolean vector b_μ on the r.h.s of (19a) only indicates which $\sigma(\mu_\ell)$ corresponding to B_{μ_ℓ} are contained in the output of the relation. They may be contained in a disjunctive or a conjunctive manner, and this fact is not given by the unary relation. In other words, we only find out, ^{e.g.} in the particular example, who are all the students who take courses with the stated set of professors ^{consisting of} A_1 and A_3 . The conjunction of the inputs A_1 and A_3 leads to concurrent the outputs the/existence of B_1 , B_3 , and B_4 . The relation will be satisfied, as will be shown in the next section on binary relations, if any one of B_1 , B_3 or B_4 is present. Therefore, the disjunction $B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_L}$ in the r.h.s of (20) is the correct description of the Boolean vector $\underline{b} = b_\mu$, $\mu = 1$ to N , i.e. the output of Eq.(18). On the other hand, it is also possible to check for the simultaneous occurrence of $B_{\mu_1}, B_{\mu_2}, \dots, B_{\mu_L}$ by taking the ^{multiple} Boolean product of b_{μ_ℓ} ($\ell = 1$ to L), if needed. Since this is only a check being done on the output vector $v(b \mu L)$, we shall indicate its use for checking the truth of the relations of the type ^{of} Eqs (1), (3), (4) and (5), after we discuss the binary relation.

(iii) BVMF equations corresponding to the clausal relation in Eq.(3)

In the notation developed above, the logical relation in Eq.(3) takes the form,

$$A_{\lambda_1} \vee A_{\lambda_2} \vee \dots \vee A_{\lambda_K} \xRightarrow{R} B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_L} \quad (23)$$

and the Boolean algebraic equations corresponding to (17-20) are summarized below, as in (24) and (25), for the following reasons.

If we consider the same problem as in subsection (ii) and wish to find out who are all the students who take courses under either professor A_1 or professor A_3 (or both), the corresponding logical expression to be adopted to test the truth of the input statement is $A_1 \vee A_3$. If either A_1 or A_3 alone is true, then the corresponding $b_{\mu}(1)$ or $b_{\mu}(3)$ would give the set of students who take their courses with the concerned professors. If, however, the only information is that either A_1 or A_3 ^{is true}, then the set of students who would take a course under a professor (A_1 or A_3) is only the union of the two sets of students described by $b_{\mu}(1)$ and $b_{\mu}(3)$. Thus, taking the particular example, this set consists of the single student B_3 who takes the course under both the professors A_1 and A_3 , so that if either is present in the subset A, then he will be taking a course with him. Therefore, the corresponding Boolean algebraic operation equivalent to the

for the relation (23),
 r.h.s of (22b),/is $(1\ 0\ 1\ 0) \otimes (0\ 0\ 1\ 1) = (0\ 0\ 1\ 0)$, which
 gives a single student B_3 who will be taking a course, ^{thus} making
 the relation ^{R to be} true, / either one of the professors A_1 or A_3
 is giving the course ^{i.e. is} (present in the subset A under consideration).
 From this example, we may generalize the corresponding formulae
 as follows.

Eq.(18) becomes Eq.(24) below.

$$\bigotimes_{\lambda=1}^M a_{\lambda} \otimes R_{\lambda\mu} = b_{\mu} \quad (24)$$

Eq.(20) becomes Eq.(25) , namely

$$\bigotimes_{k=1}^K \neg(\lambda_k) \otimes R_{\lambda_k\mu} = \bigoplus_{\ell=1}^L \neg(\mu_{\ell}) \quad (25)$$

It is obvious from the above explanation that the ^{interpretation of} logical / Eq.(25) is
 Eq.(23) giving the clausal relation between the disjunction
 of the inputs A_{λ_k} and the disjunction of the outputs $B_{\mu_{\ell}}$,
 so that Eq.(25) is the BVMF representation of Eq.(23).

We shall now consider the binary relation, namely Eq.(15b)
 and its BVMF implementation before considering the implementation
 of Eqs. (4) and (5).

(c) Binary form of implementation of the standard clausal relation

The standard clausal relation given in Eq.(1) can be expressed in the form of Eq.(21) in our notation. The relevant equation that is to be implemented is (15b) with the inputs $v(a \wedge K)$ and $v(b \vee L)$ in the equation $\underline{a} \underline{R} \underline{b} = c_\alpha$ yielding the truth value of the relation R as c_α . The corresponding BVMF formula in the Dirac bracket form is

$$\langle a | R | b \rangle = c_\alpha = \bigoplus_{\mu} ((\bigoplus_{\lambda} a_{\lambda} \otimes R_{\lambda\mu}) \otimes b_{\mu}) \quad (26)$$

Eq.(26) can be split into two parts as in (27a,b)

$$\langle a | R | = \langle b' | , \quad \langle b' | E | b \rangle = \langle b' | b \rangle = c_\alpha = t \quad (27a,b)$$

In this, (27a) is identical with the unary equation which has been discussed in the last subsection (b). However, we do not take out the output b'_{μ} of the unary relation, but check this output with the second input $v(b \vee L) \equiv b_{\mu}$, $\mu = 1$ to N by the equivalence relational matrix $|E|$ to obtain the truth value t as c_α . This will become evident if we expand $\langle b' | b \rangle$ as in (28).

$$\langle b' | b \rangle = \sum_{\mu} b'_{\mu} \otimes b_{\mu} \quad (28a)$$

$$= 1, \text{ if for some } \mu, \quad b'_{\mu} = b_{\mu} = 1 \quad (28b)$$

$$\text{and } = 0, \text{ if for no } \mu, \quad (28c)$$

b'_{μ} and b_{μ} are both equal to 1

Thus, the relation R will have a truth value $t(\underline{a}, \underline{b} | \underline{R}) = 1$ standing for "true", if the output vector of the unary relation (27a) has at least one component b'_{μ} which agrees with one component of the second input vector b_{μ} (making some $B'_{\mu\ell}$, the same as $B_{\mu\ell}$), and 0 if the two sets B' and B have no members in common, so that $b'_{\mu} = b_{\mu} = 1$ is not true for any μ , as in (28c). This agrees very well with the concept of the relation being true if there is at least one member $A_{\lambda k}$ of the set A and one member $B_{\mu\ell}$ of the set B which are related by the relational matrix element $R_{\lambda k \mu\ell} = 1$. An examination of Eq.(26) and (27) also indicates that $t = 1$ if there is one term $a_{\lambda} R_{\lambda k \mu\ell} b_{\mu} = 1$. It should be noticed that by taking the Boolean sum \bigoplus_{λ} in (26), we effectively make the check for the

disjunction of the members of the output set B. On the other hand, as explained under section (b) on the unary form of implementation, the same multiple sum \bigvee_{λ} , when applied to the product $a_{\lambda} \otimes R_{\lambda\mu}$ is effectively equivalent to the conjunction of the members A_{λ_k} in the input set A. Therefore, Eqs.(26) and (27) refer exclusively to Eq.(1), which is expressed in our notation as Eq.(21).

3. Summary of BVMF formulae for the implementation of the four types of relations.

(a) Type-1 relation

We shall call Eq.(1) as a Type-1 relation and Eqs. (3), (4) and (5) as Type-2, Type-3 and Type-4 relations. We shall give the outline of the BVMF formulae for these in this Section 3. In the subsection 3(a) we shall not only give the formulae for Type-1 relation, but also the nature of the specification of the problem, given a general matrix $R_{\lambda\mu}$ and the particular form of this matrix when the relation itself is specified by a clausal relation of Type-1. These are given in Sections A and B of Table 1. (See Section 5 for a justification of the statements in Table 1 B).

Table 1. Summary of BVMF formulae for the implementation
of a Type-1 relation

A. General relation specified by a matrix $R_{\lambda\mu}$

Nature of the relation

Logical specification: $A_{\lambda} R_{\lambda\mu} B_{\mu}$, $\lambda = 1$ to M , $\mu = 1$ to N ;

$$\begin{aligned} R_{\lambda\mu} &= 1 \text{ if } A_{\lambda} \text{ is related to } B_{\mu} \text{ and} \\ &= 0 \text{ otherwise} \end{aligned} \quad (A-1)$$

Equivalent BVMF representation:

$$\begin{aligned} \underline{a} \underline{R} \underline{b} ; \quad \underline{a} &= (a_1, a_2, \dots, a_{\lambda}, \dots, a_M), \\ \underline{b} &= (b_1, b_2, \dots, b_{\mu}, \dots, b_N); \quad \underline{R} = R_{\lambda\mu} \end{aligned} \quad (A-2)$$

B. Relational matrix specified by a clausal relation

Logical specification:

$$\begin{aligned} A_{\lambda_1^0} \wedge A_{\lambda_2^0} \wedge \dots \wedge A_{\lambda_{K^0}^0} \xrightarrow{R} B_{\mu_1^0} \vee B_{\mu_2^0} \vee \dots \vee B_{\mu_{L^0}^0} \\ \lambda_{k^0}^0 \in 1 \text{ to } M, \mu_{l^0}^0 \in 1 \text{ to } N \end{aligned} \quad (B-1)$$

BVMF representation: Same as in (A-2) except that $R_{\lambda\mu} = 1$

only if $\lambda = \lambda_{k^0}^0$, $\mu = \mu_{l^0}^0$ ($k^0 = 1$ to K^0 , $l^0 = 1$ to L^0), and $= 0$ otherwise
(B-2)

C. Implementation of a Type-1 relation R(1)1. Specification of the relationLogical formula:

$$A_{\lambda_1} \wedge A_{\lambda_2} \wedge \dots \wedge A_{\lambda_K} \xrightarrow{R} B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_L} \quad (C1-1)$$

BVM specification

$$\text{L.H.S of (C1-1)} \equiv v(a \lambda K) = \bigoplus_{k=1}^K \sigma(\lambda_k)$$

$$\text{R.H.S of (C1-1)} \equiv v(b \mu L) = \bigoplus_{l=1}^L \sigma(\mu_l) \quad (C1-2)$$

Relational matrix

$R_{\lambda \mu}$, $\lambda = 1$ to M , $\mu = 1$ to N as in Part A, or Part B, according to the nature of the problem

2. Unary form of implementation

$$\langle a | = v(a \lambda K), \quad \langle a | R | = \langle b | \mapsto \langle b | = v(b \mu L) \quad (C2-1)$$

leading to

$$\langle v(a \lambda K) | R | = \bigoplus_k \sigma(\lambda_k) \otimes R_{\lambda_k \mu} = \bigoplus_k R_{\lambda_k \mu} = b_{\mu} \quad (C2-2)$$

giving as output

$$b_{\mu} = v(b \mu L) = \bigoplus_l \sigma(\mu_l) \equiv \bigvee_l B_{\mu_l} \quad (C2-3)$$

Table 1 Contd.

3. Binary form of implementation

$$\langle a | R | = \langle b' | , \quad \langle b' | b \rangle = c_\alpha ; \quad c_\alpha = t(\underline{a}, \underline{b} | R(1)) \quad (C3-1)$$

With the inputs

$$\langle a | = v(a \lambda K), \quad \langle b' | = v(b \mu L) \quad (C3-2)$$

as in (C1-2), we have

$$\bigoplus_k (a_{\lambda_k} \otimes R_{\lambda_k \mu}) = b'_\mu ; \quad \bigoplus_\ell (b'_\mu \otimes b_{\mu_\ell}) = c_\alpha \quad (C3-3)$$

leading to

$$\bigoplus_\ell \left(\bigoplus_k a_{\lambda_k} \otimes R_{\lambda_k \mu_\ell} \right) \otimes b_{\mu_\ell} = c_\alpha = \bigoplus_{\ell=1}^L \left(\bigoplus_{k=1}^K R_{\lambda_k \mu_\ell} \right) \quad (C3-4)$$

In this table, the relation that is implemented and tested is given in Section C, which is also a Type-1 relation and having the members A_{λ_k} for the^{sub}/set A and B_{μ_ℓ} for the subset B. Since the relational matrix may be specified as in Section A, or Section B, according to the nature of the problem under consideration, the formulae in Section C for a Type-1 relation are all valid, either for a general relation $R_{\lambda\mu}$ between the members of the sets A and B as in Section A, or for the particular type of relational matrix given in Section B as demanded by a clausal relation,

The implementations of the unary and binary relations discussed in Sections 2(b) and 2(c) which are summarized in Parts C-2 and C-3 of Table 1 can be summarized still further as described in the next para.

Thus, in the unary implementation of the relation in (C1-1) given the indices λ_k of the members of the input subset A defining the input vector \underline{a} and the relational matrix $R_{\lambda\mu}$, the value of $\bigoplus_k R_{\lambda_k\mu}$ in (C2-2) can be calculated to yield the output vector $\underline{b}(\equiv b_\mu)$, which is then analysed to find out which b_{μ_ℓ} are equal to 1 and thus give the indices μ_ℓ of the output subset $B(\equiv \{B_{\mu_\ell}\})$, as in (C2-3). In the binary implementation, given the indices λ_k and μ_ℓ of the subsets A and B, and the relational matrix $R_{\lambda\mu}$, the Boolean multiple sum $\bigoplus_\ell (\bigoplus_k R_{\lambda_k\mu_\ell}) = c_\alpha$ in (C3-4) gives straightaway the truth value t or the relation in (C1-1) as the Boolean number 1 or 0.

(b) Relations of Types 2, 3 and 4

As mentioned in Section 2(b)(iii), the Type-2 relation denoted by Eq.(3) is taken to be the format of Eq.(23), with the indices λ_k and μ_ℓ , as in Table 2A Eq.(A1-1). This is the equation to be implemented, using as input the relational matrix $R_{\lambda\mu}$ as specified in Table 1A, or 1B, according to the nature of the problem.

For the unary form of implementation, the essential theory is what was developed in Section 2(b)(iii), with the only difference from Table 1, in that the input data consist of disjunctions of the input terms A_{λ_1} to A_{λ_K} , while the output is the same as for the Type-1 relation. The corresponding changes, as indicated in Eqs.(24) and (25), have been incorporated in Eqs. (A2-1), (A2-2), (A2-3) of Table 2A.

For the binary form of implementation, since the outputs for a Type-1 relation and a Type-2 relation are the same, the only change that is needed is in the unary part of implementation, namely $\langle a | \otimes | R \rangle = \langle b' |$, which replaces $\langle a | R \rangle = \langle b' |$ for the Type-1 relation. With this change, Eqs. (A3-1 to A3-4) of Table 2A readily follow from the corresponding formulae in Table 1C, namely (C3-1 to C3-4). It will be noticed that, in essence, the Boolean summation over k for the Type-1 relation is replaced by the Boolean multiple product over k in Table 2A for the Type-2 relation.

The formulae for the Type-3 relation are given in Table 2B. The logical relation is contained in Eq.(B1-1) and it differs from the Type-1 relation only in that the output is the conjunction

Table 2. Summary of BVMF formulae for implementation of relations of Types 2, 3, 4A. Type-2 relation1. Specification of the relation R(2)Logical formula

$$A_{\lambda_1} \vee A_{\lambda_2} \vee \dots \vee A_{\lambda_K} \xrightarrow{R} B_{\mu_1} \vee B_{\mu_2} \vee \dots \vee B_{\mu_L} \quad (A1-1)$$

BVM specification and relational matrix

Same as in Table 1, Section C-1 .

2. Unary form of implementation

$$\langle a | = v(a \lambda K), \quad \langle a | \otimes |R| = \langle b | \mapsto \langle b | = v(b \mu L) \quad (A2-1)$$

leading to

$$\langle v(a \lambda K) | \otimes |R| = \bigotimes_k \sigma(\lambda_k) \otimes R_{\lambda_k \mu} = \bigotimes_k R_{\lambda_k \mu} = b_{\mu} \quad (A2-2)$$

giving as output

$$b_{\mu} = v(b \mu L) = \bigoplus_l \sigma(\mu_l) \equiv \bigvee_l B_{\mu_l} \quad (A2-3)$$

3. Binary form of implementation

$$\langle a | \otimes |R| = \langle b' |, \quad \langle b' | b \rangle = c_{\alpha}; \quad c_{\alpha} = t(\underline{a}, \underline{b} | R(2)) \quad (A3-1)$$

With the inputs

$$\langle a | = v(a \lambda K), \quad \langle b | = v(b \mu L) \quad (A3-2)$$

as in (C1-2) of Table 1, we have

$$\bigotimes_k (a_{\lambda_k} \otimes R_{\lambda_k \mu}) = b'_{\mu}; \quad \bigoplus_l (b'_{\mu_l} \otimes b_{\mu_l}) = c_{\alpha} \quad (A3-3)$$

leading to

$$\bigoplus_l \left(\bigotimes_k a_{\lambda_k} \otimes R_{\lambda_k \mu_l} \right) \otimes b_{\mu_l} = c_{\alpha} = \bigoplus_{l=1}^L \left(\bigotimes_{k=1}^K R_{\lambda_k \mu_l} \right) \quad (A3-4)$$

Table 2. Contd.

B. Type-3 relation1. Logical formula of the relation R(3)

$$A_{\lambda_1} \wedge A_{\lambda_2} \wedge \dots \wedge A_{\lambda_K} \xrightarrow{R} B_{\mu_1} \wedge B_{\mu_2} \wedge \dots \wedge B_{\mu_L} \quad (B1-1)$$

2. BVM specification and relational matrix are the same as for Type-1 and Type-2 relations.

There is no unary form of implementation particularly for Type-3 relation which is different from that of Type-1 relation.

3. Binary form of implementation

$$\langle a | R | = \langle b' | , \langle b' | \otimes | b \rangle = c_\alpha ; c_\alpha = t(\underline{a}, \underline{b} | R(3)) \quad (B2-1)$$

With the same inputs as in (A3-2) above, we have

$$\bigoplus_k (a_{\lambda_k} \otimes R_{\lambda_k \mu_\ell}) = b'_{\mu_\ell} ; \bigotimes_\ell (b'_{\mu_\ell} \otimes b_{\mu_\ell}) = c_\alpha \quad (B2-2)$$

leading to

$$\bigotimes_\ell \left(\bigoplus_k a_{\lambda_k} \otimes R_{\lambda_k \mu_\ell} \right) \otimes b_{\mu_\ell} = c_\alpha = \bigotimes_{\ell=1}^L \left(\bigoplus_{k=1}^K R_{\lambda_k \mu_\ell} \right) \quad (B2-3)$$

C. Type-4 relation1. Logical formula of the relation R(4)

$$A_{\lambda_1} \vee A_{\lambda_2} \vee \dots \vee A_{\lambda_K} \xrightarrow{R} B_{\mu_1} \wedge B_{\mu_2} \wedge \dots \wedge B_{\mu_L} \quad (C1-1)$$

BVM specification and relational matrix are same as for Types-1, 2 and 3 relations.

There is no unary form of implementation for this Type 4 relation also.

2. Binary form of implementation

$$\langle a | \otimes | R | = \langle b' | , \langle b' | \otimes | b \rangle = c_\alpha ; c_\alpha = t(\underline{a}, \underline{b} | R(4)) \quad (C2-1)$$

On combining the formulae for Type-2 and Type-3 relations given in Sections A and B above, we obtain the value of c_α as

$$\bigotimes_\ell \left(\bigotimes_k a_{\lambda_k} \otimes R_{\lambda_k \mu_\ell} \right) \otimes b_{\mu_\ell} = c_\alpha = \bigotimes_{\ell=1}^L \left(\bigotimes_{k=1}^K R_{\lambda_k \mu_\ell} \right) \quad (C2-2)$$

of b_{μ_ℓ} in the Type-3 relation, unlike its being the disjunction of b_{μ_ℓ} in the Type-1 relation. Consequently, as mentioned in Section 2, the formula for unary implementation is irrelevant because this only provides the output as a disjunction of b_{μ_ℓ} . In the binary form of implementation, on the other hand, the output vector $b_{\mu_\ell}^i$ has to be checked for the simultaneous presence of the vectors $\neg(b_{\mu_\ell})$ in it, and this is done by a multiple Boolean product \bigotimes_{ℓ} as given in (B2-2) of Table 2B. Therefore, the only difference between the expression for the truth value $t = c_\alpha$ between Type-1 and Type-3 relations is the occurrence of $\bigoplus_{\ell=1}^L$ in (C3-4) of Table 1, and $\bigotimes_{\ell=1}^L$ in (B2-3) of Table 2 for Type-3 relation. The other formulae in Table 2B are quite obvious.

In the same way, the formulae for the Type-4 relation of Eq.(5), rewritten in the form of (C1-1) of Table 2C, follow from a combination of those given for the input in Type-2, and the output in Type-3, relations in Table 2 itself. The features that are different are stated in Table 2C and the binary formula for $t = c_\alpha$ contains multiple products, for both $k = 1$ to K and $\ell = 1$ to L , of $R_{\lambda_{k\mu_\ell}}$.

A comparative statement of the BVMF formulae for unary and binary forms of implementation of the four types of relations is given in Table 3. As will be seen from it, the output vector b_μ of a unary relation, and the truth value $t = c_\alpha$ of the binary form of implementation, are both expressible as Boolean sums or products of suitable $R_{\lambda\mu}$ chosen from the components of the matrix $R_{\lambda\mu}$ representing the relation.

Table 3. Summary of the BVMF formulae for the implementation of the four types of relations

Nature of the relation		BVMF formulae for implementation	
Designation	Logical connectives for $A_{\lambda k} \quad B_{\mu l}$	Unary relation $v(b^\mu L)$ $b_\mu = \bigoplus_l \neg(\mu_l)$	Binary relation $t(\underline{a}, \underline{b} R(n)) = c_\alpha$
Type-1	$\wedge \quad \vee$	$\bigoplus_k R_{\lambda k \mu}$	$\bigoplus_l (\bigoplus_k R_{\lambda k \mu l})$
Type-2	$\vee \quad \vee$	$\bigotimes_k R_{\lambda k \mu}$	$\bigoplus_l (\bigotimes_k R_{\lambda k \mu l})$
Type-3	$\wedge \quad \wedge$	—	$\bigotimes_l (\bigoplus_k R_{\lambda k \mu l})$
Type-4	$\vee \quad \wedge$	—	$\bigotimes_l (\bigotimes_k R_{\lambda k \mu l})$

A study of the results given in this section and the relationships between the formulae for these types indicate that the formulation can be extended to a completely general relation between a logical polynomial in A_{λ_k} and another logical polynomial in B_{μ_l} of the type in Eq.(29).

$$A_{\lambda_1} \vee (A_{\lambda_2} \wedge A_{\lambda_3}) \vee A_{\lambda_4} \xrightarrow{R} (B_{\mu_1} \vee B_{\mu_2}) \wedge (B_{\mu_3} \vee B_{\mu_4}) \wedge B_{\mu_5} \quad (29)$$

This is discussed in the next section.

4. BVMF formalism for a very general type of relation between logical polynomials

(a) General principles

On examining the formulae corresponding to the relations of the types 1, 2, 3, 4, which contain either conjunctions or disjunctions of the individual terms in the input or the output of the relation, it is found that they can all be derived on the basis of two simple equations governing implication involving conjunctions and disjunctions, namely

$$((A_1 \wedge A_2) \Rightarrow B) \equiv (A_1 \Rightarrow B) \vee (A_2 \Rightarrow B) \quad (30a)$$

$$((A_1 \vee A_2) \Rightarrow B) \equiv (A_1 \Rightarrow B) \wedge (A_2 \Rightarrow B) \quad (30b)$$

$$(A \Rightarrow (B_1 \wedge B_2)) \equiv (A \Rightarrow B_1) \wedge (A \Rightarrow B_2) \quad (31a)$$

$$(A \Rightarrow (B_1 \vee B_2)) \equiv (A \Rightarrow B_1) \vee (A \Rightarrow B_2) \quad (31b)$$

In essence, if the conjunction or disjunction is in the input, then the two relations on the r.h.s are connected by disjunction or conjunction respectively. On the other hand, if the conjunction or disjunction is in the output, then the two relations in the r.h.s are connected by the same connectives conjunction or disjunction as the case may be. This difference is evident in the BVMF formula in Table 3, where it will be noticed that the Boolean operation for k is the Boolean sum or Boolean product according as the logical operation for A_{λ_k} is conjunction or disjunction; while the Boolean operation is the sum or product for the output B_{μ_k} according as the logical operation is disjunction or conjunction.

The above homology between the logical operations in the description of a relation and the corresponding Boolean operations in the BVMF representation of the relation and the derived formulae for its unary and binary implementation, can be used to formulate the theory for a general relation of the type in (32), one example of which is (29).

$$\text{Type 5: } f_L(A_{\lambda_k}) \xrightarrow{R} g_L(B_{\mu_k}) \quad (32)$$

where f_L and g_L are "logical functions" of $A_{\lambda_1}, \dots, A_{\lambda_K}$

and $B_{\mu_1}, \dots, B_{\mu_L}$, containing the conjunctions, disjunctions and parantheses in any order. The function f_L or g_L , as the case may be, is specified by the positions of occurrence of these conjunctions, disjunctions, and parantheses.

We may define the corresponding Boolean functions $f_B(u_{\lambda_k})$ by replacing A_{λ_k} by u_{λ_k} , which are Boolean scalars, associated with the same indices λ_k , having the values 1 or 0, by replacing each disjunction by the Boolean sum (\oplus) and each conjunction by the Boolean product (\otimes), with the parantheses in the same positions as in $f_L(A_{\lambda_k})$; and similarly define $g_B(v_{\mu_\ell})$ by replacing B_{μ_ℓ} by v_{μ_ℓ} , which are again Boolean scalars associated with the indices μ_ℓ , and the logical operators \vee and \wedge by the Boolean operators \oplus and \otimes , keeping the parantheses in the same positions as in $g_L(B_{\mu_\ell})$. We also require, in addition, two other Boolean functions associated with $f_B(u_{\lambda_k})$ and $g_B(v_{\mu_\ell})$, denoted by $f_B^*(u_{\lambda_k})$ and $g_B^*(v_{\mu_\ell})$, which we term as the "complements" of the functions $f_B(u_{\lambda_k})$ and $g_B(v_{\mu_\ell})$. These are obtained by replacing every Boolean sum operator by a Boolean product and vice-versa, leaving the Boolean terms and the parantheses unchanged. in the expressions f_B and g_B respectively.

The relations between f_L , f_B and f_B^* and similarly between g_L , g_B and g_B^* may be illustrated by the examples from Table 2. Thus, denoting the logical formula of a Type-2 relation, namely (A1-1) of Table 2, by

$$f_L(A_{\lambda_k}) \xRightarrow{R} g_L(B_{\mu_\ell}) \quad (33)$$

the relevant part of the equation (A2-2), namely $b_{\mu} = \bigvee_k R_{\lambda_k \mu}$ can be written in this notation as

$$b_{\mu} = f_B^*(R_{\lambda_k \mu}), \quad \mu = 1 \text{ to } N \quad (34)$$

The r.h.s of Eq.(34) effectively is a set of Boolean functions, each having the Boolean scalar value 1 or 0 corresponding to b_{μ} the function itself having as its argument $R_{\lambda_k \mu}$ and being associated with f_L of (33). The star occurs in f_B^* since disjunctions in the l.h.s of (33) are replaced by Boolean products in the expression of b_{μ} in terms of $R_{\lambda_k \mu}$.

In the same way, in the binary form of implementation, we may consider Eq.(A3-3) of Table 2, namely

$$\bigoplus_{\ell} (b'_{\mu_\ell} \otimes b_{\mu_\ell}) = c_{\alpha} = \bigoplus_{\ell} c_{\alpha}(\mu_\ell) \quad (35)$$

This has a multiple sum over μ_ℓ corresponding to the multiple or

in $g_L(B_{\mu_l})$ over the same indices μ_l . Consequently, Eq.(35) can be written in the notation of the polynomial function in the form

$$c_\alpha = g_B(c_\alpha(\mu_l)) \quad (36)$$

Once again, it is to be noted that $c_\alpha(\mu_l)$ is a Boolean scalar which replaces B_{μ_l} in $g_L(B_{\mu_l})$ to give $g_B(c_\alpha(\mu_l))$.

These examples will illustrate the application of our notation and the ^{significance} and nature of the associated functions f_B^* and $*$. In what follows, we shall give a statement of the essential results for the implementation of the Type-5 relation (32) and give the proof of the formulae in the succeeding section.

(b) BVMF formulae for the unary and binary implementation of a Type-5 relation.

We shall state only the binary form of implementation since this contains two equations, one of which $\langle a | R \rangle = \langle b' |$ yields ^{the} intermediate input $\langle b' |$, which is really the output of the unary relation associated with the input $\langle a |$ and the relational matrix $R_{\lambda\mu}$. The second equation compares the $|b'\rangle$ with the ^{via} the equivalence relation $\langle b' | E | b \rangle$, second input namely $|b\rangle$ and provides the truth value of the

binary relation $\langle a | R | b \rangle$. This procedure, of effectively replacing the process of finding the truth value of a binary relation into two parts, is very vital in the formalism associated with the most general type of relation, namely Type-5. We shall therefore give the formulae, corresponding to those in Table 2A Section (A-3), Eqs (A3-1 to A3-4), for the Type-5 relation in the form (32), where the logical function is very generally defined. In the proof which will be given in the next section, we shall take the disjunctive normal form for the input and the conjunctive normal form for the output as is the case, e.g. in the relation in Eq.(29). The procedure adopted therein will indicate that this proof is equally valid also for the general polynomial form for which we give the relevant formulae below.

With the inputs

$$\langle a | = v(a \lambda K), \quad \langle b | = v(b \mu L) \quad (37a)$$

define

$$b'_{\mu}(\lambda_k) = a_{\lambda_k} \otimes R_{\lambda_k \mu}, \quad \mu = 1 \text{ to } N \quad (37b)$$

Since $a_{\lambda} = 1$ for $\lambda = \lambda_k$, we obtain $b'_{\mu}(\lambda_k) = R_{\lambda_k \mu}$,
for $\mu = 1$ to N (37c)

Then, corresponding to $f_L(A_{\lambda_k})$ in (32), the intermediate output

$$\underline{b}' = b'_{\mu} = f_B^*(b'_{\mu}(\lambda_k)) \quad (37d)$$

This can be analysed, to obtain an expansion in terms of

$\sigma(\mu_{\ell'})$ as

$$b_{\mu}^i = \bigoplus_{\ell'} \sigma(\mu_{\ell'}) \quad (37e)$$

where $\mu_{\ell'}$ are the indices of those b_{μ}^i of (37d) which are equal to 1.

Effectively, Eqs. 37(b-e) give a procedure whereby the output vector $\langle b' | = v(b' \mu L)$ can be obtained from the unary relation corresponding to (32), when applied to the input $v(a \lambda K)$ of (37a).

We shall now consider the second part of the binary form of implementation, namely

$$\langle b' | E | b \rangle = \langle b' | b \rangle = c_{\mathcal{L}} \quad (38a)$$

With $\langle b | = v(b \mu L)$ and the output represented by $g_L(B_{\mu_{\ell}})$ as in (32), we obtain $c_{\mathcal{L}}$ as follows.

Define

$$c_{\mathcal{L}}(\mu_{\ell}) = \langle b_{\mu}^i | \sigma(\mu_{\ell}) \rangle = \bigoplus_{\ell'} \langle \sigma(\mu_{\ell'}) | \sigma(\mu_{\ell}) \rangle = \bigoplus_{\ell'} b_{\mu_{\ell'}}^i \otimes b_{\mu_{\ell}} \quad (38b)$$

$$1 \text{ if some } \mu_{\ell'} = \mu_{\ell}, \ell' = 1 \text{ to } L' \text{ and } = 0 \text{ otherwise} \quad (38c)$$

Then

$$c_{\mathcal{L}} = g_B(c_{\mathcal{L}}(\mu_{\ell})) = t(f_L(A \lambda_K), g_L(B_{\mu_{\ell}}) | R(5)) \quad (38d)$$

It is to be noticed that the logical functions $f_L(A_{\lambda_k})$ and $g_L(B_{\mu_\ell})$ of (32), correspond to the Boolean functions $f_B^*(b'_\mu(\lambda_k))$ of (37d) and $g_B(c_\alpha(\mu_\ell))$ of (38d) respectively, and that the former one is the starred complementary Boolean function f_B^* corresponding to f_L , and that the latter is the non-complemented g_B corresponding to g_L , in the relation (32). The reason for this become clear in the proof that follows.

What is interesting is that, by using the BVMF formalism and separating the relation (32) into two parts — namely, a unary form of implementation to obtain b' , and an equivalence comparison check to obtain c_α — the utilizations of the functions f_L and g_L are brought into two independent steps. Consequently, the implementation of the logical polynomial function, f_L or g_L via a relation of Type 5, can be carried out in an analogous manner in Boolean algebra via f_B^* and g_B , as in the two steps of the BVMF equations in Tables 1 and 2 for Types 1-4. As mentioned earlier, we shall restrict ourselves to the standard disjunctive normal form for f_L and the conjunctive normal form for g_L in what follows, and it may also be very practical way of carrying out the calculations, since the individual steps in implementing these normal forms require only the formulae in one or the other of the four types, Type-1 to Type-4, of the logical relations.

(c) Proof of Eqs. (37) and (38)

We shall use the notation A_{λ} , B_{μ} ($\lambda = 1$ to M , $\mu = 1$ to N) to represent the members of the full sets, \mathcal{A} and \mathcal{B} , of atomic statements, and A_{λ_k} , B_{μ_ℓ} , $k = 1$ to K , $\ell = 1$ to L , $\lambda_k \in 1$ to M , $\mu_\ell \in 1$ to N , to represent the members of the polynomial describing the input and output of a relation of Type-5. However, we need one more notation to represent specifically the arrangement of the terms in the disjunctive and conjunctive normal forms. These are as follows. The input is always put in disjunctive normal form and is stated as follows.

$$f'_L(A_{\lambda_k}) = f_L(A_{\lambda_k}^r) = (A_1^1 \wedge A_2^1 \wedge \dots \wedge A_{K_1}^1) \vee (A_1^2 \wedge \dots \wedge A_{K_2}^2) \vee \dots \\ \vee (A_1^r \wedge \dots \wedge A_{K_r}^r) \vee \dots \vee (A_1^R \wedge \dots \wedge A_{K_R}^R) \quad (39)$$

A general A_{λ_k} is replaced by $A_{\lambda_k}^r$ in (39). (For practical purposes, one could associate the two indices (r, k_r) with the corresponding index λ_k with $k = 1$ to K .) In the same way, the output statement is always put in the conjunctive normal form (40) with similar notation, namely,

$$g'_L(B_{\mu_\ell}) = g_L(B_{\mu_\ell}^s) = (B_1^1 \vee B_2^1 \vee \dots \vee B_{L_1}^1) \wedge \dots \\ \wedge (B_1^s \vee \dots \vee B_{L_s}^s) \wedge \dots \wedge (B_1^S \vee \dots \vee B_{L_S}^S) \quad (40)$$

where a general B_{μ_ℓ} is replaced by $B_{\mu_\ell}^s$ and one can associate

the pair of indices (s, ℓ_s) with a corresponding μ_ℓ ,
 $\ell = 1$ to L .

We further assume that the Boolean values of $R_{\lambda\mu}$,
 $\lambda = 1$ to M , $\mu = 1$ to N , are available. Then the two steps
in the binary form of implementation as outlined in (37) and (38)
can be proved as follows.

We may write the general function $f_L(A_{k_r}^r)$ as the disjunction
of a set of R functions $f_L^r(A_{k_r}^r)$ as in (41a,b):

$$f_L(A_{k_r}^r) = \bigvee_{r=1}^R f_L^r(A_{k_r}^r) ; \quad f_L^r(A_{k_r}^r) = \bigwedge_{k_r=1}^{K_r} A_{k_r}^r \quad (41a,b)$$

In this, each $f_L^r(A_{k_r}^r)$ has the expression

$$f_L^r(A_{k_r}^r) = A_1^r \wedge A_2^r \wedge \dots \wedge A_{K_r}^r \quad (41c)$$

which is a conjunction of K_r terms corresponding to the standard
form of input for the Type-1 relation.

In a similar way, we can express $g_L(B_{\ell_s}^s)$ as a conjunction
of S functions $g_L^s(B_{\ell_s}^s)$ as in (42a,b).

$$g_L(B_{\ell_s}^s) = \bigwedge_{s=1}^S g_L^s(B_{\ell_s}^s) ; \quad g_L^s(B_{\ell_s}^s) = \bigvee_{\ell_s=1}^{L_s} B_{\ell_s}^s \quad (42a,b)$$

in which each $g_L^s(B_{\ell_s}^s)$ has the expansion

$$g_L^s(B_{\ell_s}^s) = B_1^s \vee B_2^s \vee \dots \vee B_{L_s}^s \quad (42c)$$

which is a disjunction of L_s terms corresponding to the standard form of the output of the Type-1 relation.

The way in which the binary relation is implemented via two steps, the first of which is the implementation of the unary relation connecting the input $f_L^r(A_{k_r}^r)$ with the intermediate output B_{μ}^r , and the second an equivalence relation between B_{μ}^r and $g_L^s(B_{\ell_s}^s)$, may be outlined as follows.

(i) Steps involving unary form of implementation

We may combine (41a,b) and make use of the fact that, in the latter, $f_L^r(A_{k_r}^r)$ is a conjunction of K_r terms as in a Type-1 relation, and that the latter is a disjunction of R terms $f_L^r(A_{k_r}^r)$, so that by employing the first half of Eq.(C3-3) of Table 1 for Type-1 relation, and the first half of Eq.(A3-3) in Table-2 for the Type-2 relation, we obtain

$$\bigoplus_{k_r} (a_{k_r}^r \otimes R_{\lambda_{k_r}^r \mu}) = b_{\mu}^r = \bigoplus_{k_r} R_{\lambda_{k_r}^r \mu} \quad (43a)$$

$$\bigotimes_r b_{\mu}^r = b_{\mu}^r = \bigotimes_r \left(\bigoplus_{k_r} R_{\lambda_{k_r}^r \mu} \right) \quad (43b)$$

In this $\lambda_{k_r}^r$ is a function of two indices r and k_r . In terms of these same two indices, the expression for $f_L(A_{k_r}^r)$ takes the form

$$f_L(A_{k_r}^r) = \bigvee_{r=1}^R \left(\bigwedge_{k_r=1}^{K_r} A_{k_r}^r \right) \quad (43c)$$

On comparing (43b,c), it will be noticed that the multiple or in (43c) is replaced by the multiple Boolean product over r and the multiple and over k_r is replaced by the multiple Boolean sum over the same index k_r . Hence we may write

$$b_\mu' = f_B^*(R \lambda_{k_r}^r \mu) \quad (43d)$$

For utilizing these components b_μ' of the intermediate output \underline{b}' of the unary part of the binary relation, we may express it as a sum of N-vectors $\sigma(\mu_{\ell'})$, as in (44) below,

$$b_\mu' = \bigoplus_{\ell'} \sigma(\mu_{\ell'}) \quad (44)$$

where $\mu_{\ell'}$ denotes those values of μ for which $b_\mu' = 1$.

(ii) Second part of calculating the binary truth value

In an exactly similar manner, we make use of the fact that $g_L(B_{\ell_s}^s)$ is expressible in a form analogous to (43c) as

$$g_L(B_{\ell_s}^s) = \bigwedge_{s=1}^S \left(\bigvee_{\ell_s=1}^{L_s} B_{\ell_s}^s \right) \quad (45a)$$

For evaluating the value of $c_{\mathcal{L}}$ corresponding to the Boolean vector-matrix product

$$c_{\mathcal{L}} = \langle b'_{\mu} | E | g_L(B_{\ell_s}^s) \rangle \quad (45b)$$

we make use of the second part of Eq.(A3-3) of Table 2, for Type-2 relation, and of Eq. (B2-2) of Table 2, for Type-3 relation, corresponding respectively to the disjunction of the individual terms of the output and the conjunction of such terms respectively. Then we obtain,

$$\bigoplus_{\ell_s} (b'_{\mu_{\ell_s}} \otimes b_{\mu_{\ell_s}^s}) = c_{\mathcal{L}}^s = \bigoplus_{\ell_s} (b'_{\mu_{\ell_s}^s} \otimes b_{\mu_{\ell_s}^s}) \quad (45c)$$

and

$$\bigotimes_s c_{\mathcal{L}}^s = c_{\mathcal{L}} = \bigotimes_s \left(\bigoplus_{\ell_s} b'_{\mu_{\ell_s}^s} \otimes b_{\mu_{\ell_s}^s} \right) \quad (45d)$$

On comparing (45a) with (45d) it is clear that the former can be expressed in the form

$$c_{\mathcal{L}} = g_B(b'_{\mu_{\ell_s}^s} \otimes b_{\mu_{\ell_s}^s}) \quad (45e)$$

Thus, for a general relation of the Type 5, with f_L being in the disjunctive normal form and g_L being in the conjunctive normal form, we obtain both the unary relation implemented in terms of f_B^* , and the binary relation implemented in terms of

f_B^* and f_L , as in (43d) and (45e). On comparing these two equations with (37d) and (38d), and replacing $\lambda_{k_r}^r$ by λ_k as mentioned in connection with Eq.(39), and similarly $\mu_{\ell_s}^s$ by μ_ℓ it will be seen that Eqs. (43d) and (45e) are only special cases of Eqs (37d) and (38d) when f_L and g_L have the disjunctive normal form and the conjunctive normal form respectively.

However, the method of proof employed indicates that conjunctions and disjunctions may occur in any order with suitable parantheses and the formal procedures, of obtaining f_B^* for the unary relation from f_L , and g_B for the second part of the binary relation from g_L , are still maintained, so that the general equations given in (37) and (38) will continue to be valid.

5. An interesting relation to quantifier states of the input and output of relations of Types 1 to 4.

It may be pointed out that the two forms of input and output in which the individual terms are connected by conjunctions and disjunctions respectively, can be connected up with the quantifier states $(q_a k)(A_{\lambda_k})$ and $(q_b \ell)(B_{\mu_\ell})$ of the subset of terms $\{A_{\lambda_k}\}$, $k = 1$ to K , and $\{B_{\mu_\ell}\}$, $\ell = 1$ to L associated

with these two quantities. Thus the following definitions of the two standard quantifier states $(\forall k)$ and $(\exists k)$ (as developed in [6]) is quite obvious.

$$(\forall k)(A_{\lambda_k}) = \bigwedge_k A_{\lambda_k} \quad ; \quad (\exists k)(A_{\lambda_k}) = \bigvee_k A_{\lambda_k} \quad (46a,b)$$

$$(\forall \ell)(B_{\mu_\ell}) = \bigwedge_\ell B_{\mu_\ell} \quad ; \quad (\exists \ell)(B_{\mu_\ell}) = \bigvee_\ell B_{\mu_\ell} \quad (46c,d)$$

Then, the Types 1 to 4 of the relations discussed in this paper can be restated as in (47a-d).

$$\text{Type-1 : } (\forall k)(A_{\lambda_k}) \xrightarrow{R} (\exists \ell)(B_{\mu_\ell}); \text{ Type-2: } (\forall k)(A_{\lambda_k}) \xrightarrow{R} (\forall \ell)(B_{\mu_\ell}) \quad (47a,b)$$

$$\text{Type-3 : } (\exists k)(A_{\lambda_k}) \xrightarrow{R} (\exists \ell)(B_{\mu_\ell}); \text{ Type-4: } (\exists k)(A_{\lambda_k}) \xrightarrow{R} (\forall \ell)(B_{\mu_\ell}) \quad (47c,d)$$

The truth value $t_n \equiv t(R(n))$, as a Boolean number 1 or 0, gives essentially the existence of the relation $(A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell})$ when the input and the output of the relation has the quantifier states $(q_a k)$ and $(q_b \ell)$ over the subsets A_{λ_k} and B_{μ_ℓ} respectively.

Since the variables k and ℓ in Eqs. (47a-d) are independent, the QL-2 relation in each of them can be converted into its equivalent in the QL-1 form as given in (48a-d).

$$\text{Type-1} : (\exists k)(\exists \ell) (A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) \quad (48a)$$

$$\text{Type-2} : (\exists k)(\forall \ell) (A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) \quad (48b)$$

$$\text{Type-3} : (\forall k)(\exists \ell) (A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) \quad (48c)$$

$$\text{Type-4} : (\forall k)(\forall \ell) (A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) \quad (48d)$$

(a) Significance of the binary truth value t_n

These equations provide a direct alternative derivation of the BVMF formulae for $t(R(n)) = c_\alpha$ in the four rows of column 4 of Table 3. The quantifiers $(\forall k)$ and $(\exists k)$ are replaced by the BA-1 equivalents \bigotimes_k and \bigoplus_k for the logical connectives \bigwedge_k , \bigvee_k which are contained in (46a,b) defining these quantifiers, and similarly for $(\forall \ell)$ and $(\exists \ell)$. Then the four BVMF equations, mentioned above, in Table 3 are equivalent to the four equations in (48a-d) for the Types 1 to 4 of relations considered in this report. It is to be noted, however, that the Boolean operator corresponding to the quantifier $(q_b \ell)$ occurs to the right of the one for $(q_a k)$ since logical equations are written from left to right, while in matrix algebra, the application of the operation \bigoplus_ℓ or \bigotimes_ℓ occurs to the left of the corresponding operation \bigoplus_k or \bigotimes_k as the case may be. The exact

interpretation of Eqs. (48a-d) corresponding to the truth values $t(R(n))$ given in Tables 1, 2 and 3 may be given as follows. Considering the Type-1 relation first, the value of $t_1 = \bigoplus_{\ell} \left(\bigoplus_k R_{\lambda_k \mu_{\ell}} \right)$ equal to 1 or 0, is equivalent to saying that the multiply quantified statement in (48a), namely $(\exists k)(\exists \ell)(A_{\lambda_k} R_{\lambda_k \mu_{\ell}} B_{\mu_{\ell}})$ is true, or false. It will be noticed that we are given the nature of the relation in the form of the elements of the Boolean matrix $R_{\lambda \mu}$ being equal to 1 or 0. We are further given (in the Type-1 relation in (47a)) that all members of A_{λ_k} are present, and we seek to find out which members of the set $B_{\mu_{\ell}}$ are present if they are related by $R_{\lambda \mu}$ in the unary form of implementation. In the binary form, we ask whether the relation is true; and, as we have shown above, the relation is true for (48a) if there is at least one k and one ℓ for which $R_{\lambda_k \mu_{\ell}}$ is equal to 1, so that there exists a $B_{\mu_{\ell}}$ which is related to one of the A_{λ_k} . In the same way, the value of $t_2 = \bigoplus_{\ell} \left(\bigotimes_k R_{\lambda_k \mu_{\ell}} \right)$ is "true", if for all A_{λ_k} , there is at least one $B_{\mu_{\ell}}$ which is related to all of the A_{λ_k} . Similar considerations hold for $t_3 = 1$ and $t_4 = 1$.

On the other hand, $t_1 = 0$ negates the multiple quantifier $(\exists k)(\exists \ell)$ in (48a) which yields

$$\begin{aligned} \neg(\exists k)(\exists \ell)(A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) &= (\neg(\exists k) \vee \neg(\exists \ell))(A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell}) \\ &= (\forall k) [\neg(A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell})] \vee (\forall \ell) [\neg(A_{\lambda_k} R_{\lambda_k \mu_\ell} B_{\mu_\ell})] \quad (49a, b) \end{aligned}$$

This means that either no A_{λ_k} exists which is related to B_{μ_ℓ} , or no B_{μ_ℓ} exists that is related to an A_{λ_k} . If either is known to be not true, then the other one follows.

The above considerations can give the answers to questions in predicate calculus for sets of terms related by a clausal relation of any one of the four types, which can be worked out practically by a computer using the Boolean algebraic formalism developed here. The question may now be asked as to what is the nature of the quantifiers $(q_a k)$ or $(q_b \ell)$ if we combine two or more of the truth values t_1 to t_4 . We shall briefly examine this and point out a very interesting consequence regarding the quantifier states that emerges. A full discussion is reserved for a separate report.

(b) Determination of quantifier states from truth values of relations

It is possible to use the information regarding the quantifier state $(q_a k)(A_{\lambda k})$ of the input of a relation of Types 1-4 to obtain the truth value of the quantifier of the output, namely $(q_b \ell)(B_{\mu \ell})$. Thus, suppose it is known that $(\forall k)(A_{\lambda k})$ is T, then the truth values t_1 and t_2 , of the relation of Types 1 and 2, will depend essentially on the quantifier state of the output — which is $(\exists \ell)(B_{\mu \ell})$ for the Type-1 relation and $(\forall \ell)(B_{\mu \ell})$ for the Type-2 relation. Consider now the four combinations possible for (t_1, t_2) , namely $(1, 1)$, $(1, 0)$, $(0, 0)$, $(0, 1)$. Then the relation $A_{\lambda} R_{\lambda \mu} B_{\mu}$ is effectively true for the quantifier state (50) of the relation, namely,

$$(\exists k)(q_b \ell)(A_{\lambda k} R_{\lambda k \mu \ell} B_{\mu \ell}) \quad (50)$$

in which $(q_b \ell)$ has the quantifier state as given by (51a-d), for the four combinations of (t_1, t_2) .

$$t_1 = 1, t_2 = 1 \implies (q_b \ell)(B_{\mu \ell}) = (\forall \ell)(B_{\mu \ell}), \text{ since } (\forall \ell)(B_{\mu \ell}) \text{ given by } t_2 = 1 \text{ implies } (\exists \ell)(B_{\mu \ell}) \text{ required by } t_1 = 1 \quad (51a)$$

$$t_1 = 1, t_2 = 0 \implies (q_b \ell)(B_{\mu \ell}) = (\exists \ell)(B_{\mu \ell}) \wedge \neg(\forall \ell)(B_{\mu \ell}) \quad (51b)$$

since the two pieces of information are coexistent*

* We have used \wedge in this as in standard logic ; strictly the operator involved is \otimes in BVMF .

$$\begin{aligned}
t_1 = 0, t_2 = 0 &\mapsto (q_b \ell)(B_{\mu \ell}) = \neg(\exists \ell)(B_{\mu \ell}), \text{ since} \\
\neg(\exists \ell)(B_{\mu \ell}) &\equiv (\forall \ell)(\neg B_{\mu \ell}) \text{ given by } t_1 = 0 \text{ implies} \\
\neg(\forall \ell)(B_{\mu \ell}) &\equiv (\exists \ell)(\neg B_{\mu \ell}) \text{ demanded by } t_2 = 0 \quad (51c)
\end{aligned}$$

$$\begin{aligned}
t_1 = 0, t_2 = 1 &\text{ is impossible, since } \neg(\exists \ell)(B_{\mu \ell}) \text{ required} \\
\text{by } t_1 = 0 &\text{ contradicts } (\forall \ell)(B_{\mu \ell}) \text{ given by } t_2 = 1 \quad (51d)
\end{aligned}$$

It is extremely interesting that the only three quantifier states that are obtained for q_b corresponding to $(q_b \ell)(B_{\mu \ell})$, are the three generator states $\forall, \exists, \emptyset$ respectively of the BA-3 algebra of quantifier logic (QBA) developed in [1], [2], [6] from the BVMF approach. This will be pursued further in a separate report.

References

- 1, 2 . Ramachandran, G.N., "Vector-matrix representation of Boolean algebras and application to Extended Predicate Logic (EPL)", Parts I and II, Curr. Sci, 52 , p 292, 335 (1983).
- 3, 4 . Ramachandran, G.N., "New hardware circuits for the implementation of logical relations in information processing", Parts I and II, Curr. Sci., 55 , p 12, 67 (1986)
5. Kowalski, R., "Logic for Problem Solving", Elsevier-North Holland, New York (1979).
6. Ramachandran, G.N., "BVMF theory of quantifiers with BA-1 and BA-2 truth values", Matphil Reports No. 52, p 1-45 (1986).
7. Ramachandran, G.N., "BVM formulation of First Order Predicate Logic", Matphil Reports No. 53, p. 1-125 (1986)
8. Ramachandran, G.N., "BVM formalism for QL-1 in First Order Predicate Logic", Matphil Reports No. 54, p. 1-113 (1986).

SYMMETRY PROPERTIES OF LOGICAL FUNCTIONS IN PROPOSITIONAL CALCULUS

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

SYMMETRY PROPERTIES OF LOGICAL FUNCTIONS IN PROPOSITIONAL CALCULUS

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Preface

This report consists of three notes, N-1, N-2 and N-3, which arose as a consequence of developing the ideas in MR-57. They can be read in succession and the notation is consistent with the recent lecture notes MR-55 and 56.

N-1 is an extension of the four operators \underline{E} , \underline{L} , \underline{M} , \underline{N} which are applicable to quantifier states, to general logical functions $f(ai)$ in BA-1, employing the two standard logical connectives \wedge and \vee . Here also, four functions derivable from $f(ax)$, namely $f^e(ax)$, $f^l(ax)$, $f^m(ax)$, $f^n(ax)$, are defined in exactly the same manner as for quantifiers. The four operators \underline{E} , \underline{L} , \underline{M} , \underline{N} form a group exactly as for quantifier states and these related functions are very useful for a rigorous theory of quantifier states employing BA-1 truth values, which is developed in N-2.

The note N-3 is a still further generalization of N-2, employing BA-2 truth values. As shown in MR-52, the eight quantifier states of QPL employing BA-3 algebra, which are obtainable even from the use of BA-1 truth values for ax , are left unchanged on going over to BA-2 truth values \underline{ax} ,

by suitably defining the function $f(\underline{a}x)$ corresponding to each of the eight quantifier states.

It is believed that these three notes form a very satisfactory way of defining the whole of the quantifier algebra in terms of functions of logical terms employing the connectives of propositional calculus.

CONTENTS

	<u>Page No</u>
<u>N - 1 : Generalized BVMF theory</u>	
1. Extension of the relations! between the four standard quantifiers to logical functions $f_L(a_i)$	1
2. Basic transformation equations between QL-1 and QL-2 forms of elementary relations involving quantifiers	3
 <u>N - 2 : Rigorous theory of quantifier states in logic</u>	
1. Theory employing BA-1 representation for individual terms	6
2. Derivation of QL-1,2 transformations discussed in N-1	11
 <u>N - 3 : Theory of quantifiers employing BA-2 algebra for truth values</u>	
1. Extension of Eqs. (1a,b,c) of N-2 to BA-2 truth values	14
2. Need for Boolean connectives \underline{U} and \underline{V} in addition to matrix connectives \underline{A} and \underline{O}	19
3. Boolean algebraic representation of connectives in predicate logic	21
4. Non-distributivity of logical functions	23
5. Symmetry properties of quantifier states and their connectives	29

Generalized BVMF theory

1. Extension of the relations between the four standard quantifiers to logical functions $f_L(a_i)$

The four standard quantifier states $\forall, \exists, \bigwedge, \bigoplus$ have the relationships of the type

$$q(i) = q(i^e), q(i^l), q(i^m), q(i^n) = \underline{q}(i) \underline{E}, \underline{q}(i) \underline{L}, \underline{q}(i) \underline{M}, \underline{q}(i) \underline{N} \quad (1)$$

In the same way, we can define not only the function $f_L^*(a_i)$ associated with the function $f_L(a_i)$, but also the functions $\neg f_L(a_i)$ and $\neg f_L^*(a_i)$ having the forms as defined below.

Taking the quantifier state $(\forall x)(ax) = \bigwedge_x ax = q(1ax)$, the four standard quantifiers in Eq.(1) take the forms

$$(\forall x)(ax) = \bigwedge_x ax = q(1ax), i = 1 \quad (2a)$$

$$(\exists x)(ax) = \bigvee_x ax = q(i^l ax) \quad (2b)$$

$$\neg(\forall x)(ax) = (\bigwedge x)(ax) = \neg(\bigwedge_x ax) = \bigvee_x (\neg ax) = q(i^m ax) \quad (2c)$$

$$\neg(\exists x)(ax) = (\bigoplus x)(ax) = \neg(\bigvee_x ax) = \bigwedge_x (\neg ax) = q(i^n ax) \quad (2d)$$

In this the logical expressions on the r.h.s may be replaced by the form $f_L(ax) = \bigwedge_x ax$ when they take the formal descriptions as in (3) below:

$$\begin{aligned} (\forall x)(ax) &= f_L(ax) ; \quad (\exists x)(ax) = f_L^l(ax) = f^*(ax) ; \\ (\bigwedge x)(ax) &= f^m(ax) = \neg f(ax) ; \quad (\bigoplus x)(ax) = f^n(ax) = \neg f^*(ax) \end{aligned} \quad (3a,b,c,d)$$

In these equations the particular form of $f_L(ax)$ given in Eq.(3) for the quantifiers can be generalized to any logical function $f_L(ax)$ containing the logical connectives \neg, \wedge, \vee . The superscript symbol ℓ interchanges the "and"s and "or"s. The superscript symbol n converts every ax into $\neg ax$, and the superscript symbol m interchanges and's and or's and also converts each ax into $\neg ax$, effectively leading to the negation of $f_L(ax)$.

As an example, we may put $i = 6$ standing for $(\exists x)(ax)$. The same operations of ℓ, m, n , will produce in sequence $(\exists x)(ax), (\forall x)(ax), (\exists x)(\neg ax)$ and $(\forall x)(\neg ax)$. With these stipulations, all functions of quantified terms, and equations involving them, can also be treated in a similar manner. Thus the proof of $(\exists x)(ax) = \neg(\forall x)(\neg ax)$ is given by the / De-Morgan relation
$$\bigvee_x ax = \neg(\bigwedge_x (\neg ax)).$$
 The most interesting application is in what has been described as the QL-1,2 and QL-2,1 transformations in MR-53 and 54.

In fact, a very general formulation of quantifier algebra in terms of logical functions has been formulated and is being worked out. Some of the essential features will be discussed in the next few days.

2. Basic transformation equations between QL-1 and QL-
forms of elementary relations involving quantifiers

$$(a) \quad (\forall x)(\underline{ax} \Rightarrow \underline{b}) = (\exists x)(\underline{ax}) \Rightarrow \underline{b}, \quad (\exists x)(\underline{ax} \Rightarrow \underline{b}) = (\forall x)(\underline{ax}) \Rightarrow \underline{b}$$

For proving this, we ^{employ} the results given in Eqs. (4b) and (4c) which follow from the De Morgan relations for multiple conjunctions and multiple disjunctions given in (4a) :

$$\begin{aligned} (\underline{a_1} \Rightarrow \underline{b}) \wedge (\underline{a_2} \Rightarrow \underline{b}) &= (\neg \underline{a_1} \vee \underline{b}) \wedge (\neg \underline{a_2} \vee \underline{b}) \\ &= (\neg \underline{a_1} \wedge \neg \underline{a_2}) \vee \underline{b} = \neg(\underline{a_1} \vee \underline{a_2}) \vee \underline{b} = (\underline{a_1} \wedge \underline{a_2}) \Rightarrow \underline{b} \end{aligned} \quad (4a)$$

$$\bigwedge_x (\underline{ax} \Rightarrow \underline{b}) = (\bigvee_x \underline{ax}) \Rightarrow \underline{b} \quad (4b)$$

$$\bigvee_x (\underline{ax} \Rightarrow \underline{b}) = (\bigwedge_x \underline{ax}) \Rightarrow \underline{b} \quad (4c)$$

On substituting for the multiple or and multiple and , the quantifier equivalences in (2a,b), we obtain the two equations to be proved.

$$(b) \quad (\forall x)(\underline{ax} \wedge \underline{b}) = (\forall x)(\underline{ax}) \wedge \underline{b}, \quad (\exists x)(\underline{ax} \wedge \underline{b}) = (\exists x)(\underline{ax}) \wedge \underline{b}$$

This equation follows from Eqs. (5a,b)

$$\bigwedge_x (\underline{ax} \wedge \underline{b}) = \bigwedge_x (\underline{ax}) \wedge \underline{b} \quad (5a)$$

$$\bigvee_x (\underline{ax} \wedge \underline{b}) = \bigvee_x (\underline{ax}) \wedge \underline{b} \quad (5b)$$

$$(c) \quad (\forall x)(\underline{ax} \vee \underline{b}) = (\forall x)(\underline{ax}) \vee \underline{b}, \quad (\exists x)(\underline{ax} \vee \underline{b}) = (\exists x)(\underline{ax}) \vee \underline{b}$$

This also follows from the two equations corresponding to (5a,b) obtained by replacing the conjunctions in (b) by disjunctions in (c).

.4.

N-1
30.1.87

$$(d) (\sum x)(\underline{ax} \wedge \underline{b}) = (\sum x)(\underline{ax}) \wedge \underline{b} ; (\sum x)(\underline{ax} \vee \underline{b}) = (\sum x)(\underline{ax}) \vee \underline{b}$$

This will be proved from apriori considerations from the definition of $(\sum x)(ax)$ as in (6a,b)

$$(\sum x)(\underline{ax}) \bigvee_{x=1}^n (\underline{ax}) \equiv (\sum x)(\neg ax) = \bigvee_{x=1}^n (\neg \underline{ax}) \quad (6a,b)$$

where n can have any value between 1 and $N-1$, N being the number of members (ax) under considerations. Then, from (6a), we obtain

$$(\sum x)(\underline{ax}) \wedge \underline{b} = \bigvee_1^n (\underline{ax}) \wedge \underline{b} = \bigvee_1^n (\underline{ax} \wedge \underline{b}) = (\sum x)(\underline{ax} \wedge \underline{b}) \quad (7a,b,c,d)$$

It is to be noted that the multiple summation 1 to n is the same for the l.h.s (7b) and r.h.s (7c), and therefore n can have any value between 1 and $N-1$ in (7c) also, which is the very definition of the quantifier state \sum . The justification in the shift of the bracket between (7b) and (7c) is obtainable by the fact that the operations of conjunction and disjunction in Boolean algebra are commutative, because of which Eqs. (8a,b) follow:

$$\begin{aligned} (\underline{a}_1 \wedge \underline{b}) \vee (\underline{a}_2 \wedge \underline{b}) &= (\underline{a}_1 \vee \underline{a}_2) \wedge \underline{b} ; \\ (\underline{a} \wedge \underline{b}_1) \vee (\underline{a} \wedge \underline{b}_2) &= \underline{a} \wedge (\underline{b}_1 \vee \underline{b}_2) \end{aligned} \quad (8a,b)$$

from which (8c) and its counterpart (8d) follow:

$$(\underline{a}_1 \vee \underline{b}) \vee (\underline{a}_2 \vee \underline{b}) \vee \dots \vee (\underline{a}_n \vee \underline{b}) = (\underline{a}_1 \wedge \underline{a}_2 \wedge \dots \wedge \underline{a}_n) \vee \underline{b} \quad (8c)$$

$$(\underline{a} \wedge \underline{b}_1) \vee (\underline{a} \wedge \underline{b}_2) \vee \dots \vee (\underline{a} \wedge \underline{b}_n) = \underline{a} \wedge (\underline{b}_1 \vee \underline{b}_2 \vee \dots \vee \underline{b}_n) \quad (8d)$$

In the same way, the second equation under (c) can be proved.

$$(e) \quad \underline{(\ominus x)(\underline{ax} \wedge \underline{b})} = \underline{(\ominus x)(\underline{ax}) \wedge \underline{b}} ; \underline{(\ominus x)(\underline{ax} \vee \underline{b})} = \underline{(\ominus x)(\underline{ax}) \vee \underline{b}}$$

$$(f) \quad \underline{(\underline{\zeta} x)(\underline{ax}) \Rightarrow \underline{b}} = \underline{(\ominus x)(\underline{ax}) \Rightarrow \underline{b}} ; \underline{(\ominus x)(\underline{ax} \Rightarrow \underline{b})} = \underline{(\underline{\zeta} x)(\underline{ax}) \Rightarrow \underline{b}}$$

All the equations in (a) to (f), as well as the corresponding ones for the logical connectives $\wedge, \vee, \Rightarrow$ between \underline{a} and \underline{bx} will be proved as the consequences of a very general set of equations (9a,b,c) and (10), which we shall prove namely

$$\begin{aligned} \underline{(\underline{qx})(\underline{ax}) \wedge \underline{b}} &= \underline{(\underline{qx})(\underline{ax} \wedge \underline{b})} ; \underline{(\underline{qx})(\underline{ax}) \vee \underline{b}} = \underline{(\underline{qx})(\underline{ax} \vee \underline{b})} ; \\ \underline{(\underline{qx})(\underline{ax}) \Rightarrow \underline{b}} &= \underline{(\underline{q}^b x)(\underline{ax} \Rightarrow \underline{b})} \end{aligned} \quad (9a,b,c)$$

$$\underline{a} \underline{z}(\underline{qy})(\underline{by}) = (\underline{qy})(\underline{a} \underline{z} \underline{by}), \quad \underline{z} = \wedge, \vee, \Rightarrow \quad (10)$$

these
In / $\underline{(\underline{qx})}$ and $\underline{(\underline{qy})}$ can be any one of the eight quantifiers

in BA-3 algebra . This is shown in the next note N-2.

Rigorous theory of quantifier states in logic1. Theory employing BA-1 representation for individual terms

We assume that ax can only have two states T and F designated by the Boolean scalars 1 and 0. Then we define the quantifier state $q(ax)$ by Eqs (1a,b,c):

$$q(ax) = \bigvee_{m=N_1}^{N_2} f(ax, m, n), \quad m+n=N, \quad (1a)$$

where

$$f(ax, m, n) = \bigvee_{j=1}^{N_{cm}} f_j(ax, m, n) \quad (1b)$$

and

$$f_j(ax, m, n) = \bigwedge_{r=0}^m (ax_r) \bigwedge_{s=0}^n (\neg(ax_s)), \quad m+n=N \quad (1c)^\dagger$$

and there are $j = 1$ to N_{cm} combinations (m, n) for x_r and x_s . A quantifier state is the truth value of $q(ax)$ in (1a), given the individual truth values 1 or 0 of x_r and x_s in (1c), which are m and n in number*. The range of m , namely N_1 to N_2 , specifies the nature of the quantifier. The value of N_1, N_2 for the four standard quantifier states $\forall, \exists, \bigwedge, \bigoplus$ are given in the top half of Table 1. They agree completely with the standard concepts of quantifiers, namely that $\forall(ax)$ is true only if n , the number of negated terms is nil, and similarly $\bigoplus(ax) = \forall(\neg ax)$ is true only if m , the number affirmative terms is 0. These are the only two singular states defined by a unique value for the range N_1 to N_2 .

*In (1a) and (1b), we make use of the fact that the multiple or in $t_{N_1} \vee t_{N_1+1} \vee \dots \vee t_{N_2} = T$ if at least one of them is true.

†The multiple conjunction is to be taken to have the truth value T if either m or n is equal to 0.

1a.

N-2
2.2.87

Table 1. Description via set theory of the quantifier states
q(1) to q(8) of QBA

$i =$	$q(iax)$	$m =$ $N_1 \quad N_2$		$n =$ $N'_1 \quad N'_2$	
1	\forall	N	N	0	0
6	\exists	1	N	N-1	0
6	\wedge	0	N-1	N	1
5	Φ	0	0	N	N
3	Σ	1	N-1	N-1	1
7	Δ	0	N	N	0
4	\ominus	N	N	0	0
		or 0	0	N	N
8	\emptyset	—	—	—	—

.2.

N-2
3.2.87
Draft-2

The definition of the quantifier state \exists is usually given in terms of the operator \bigvee . However, it can also be given a definition in terms of Eqs.(1a,b,c), where the range of m can be anywhere from 1 to any value N . This means that $f(ax, m, n)$ will be true for at least one value of m and it can also be true for all the N -values of m , including $n = N$. This definition agrees with the standard definition of $\exists(ax)$

$$\exists(ax) = \bigvee_{m=1}^N (ax) \quad (3a)$$

as in (2b) of N-1, and it also agrees with our definition of $\bigvee(\neg ax)$ in terms of which we have Eq.(3b).

$$\neg \bigvee(\neg ax) = \bigwedge_{m=1}^N (\neg ax) = \exists(ax) \quad (3b)$$

In the same way, the quantifier state $\bigwedge(ax) \equiv \exists(\neg ax)$ has the range for m from 0 to $N-1$, which makes the range of n from 1 to N , and is expressible as $\bigvee_{n=1}^N (\neg ax)$.

In QBA we have the generator states as \bigvee , \bigoplus and \bigwedge where the range of m for \bigwedge is the intersection of the ranges for m of \exists and \bigwedge , namely 1 to $N-1$. This can also be given

.3.

N-2
3-2-87
Draft-2

a straightforward definition in terms of Eqs.(1a) and (1b), by saying that m cannot correspond to "All" or "None", but only to "some"; making $N_1 = 1$ and $N_2 = (N - 1)$, in 1(a). Of the remaining three quantifier states in QBA Θ , Δ and ϕ , which find expression in terms of disjunction or conjunction of two or three of the generator states \forall , Φ , Σ , as in (4a,b,c), Δ and ϕ also have a unique representation in the form of Eq.(1) as shown in Table 1.*

$$q(4) = \Theta = \forall \quad \Phi = q(1) \vee q(5) \quad (4a)$$

$$q(7) = \Delta = \forall \quad \Sigma \oplus \Phi = q(1) \vee q(3) \vee q(5) \quad (4b)$$

$$q(8) = \phi = \forall \quad \Sigma \otimes \Phi = q(1) \wedge q(3) \wedge q(5) \quad (4c)$$

On the other hand, Θ is not expressible by a single continuous range of values for m . It requires the disjunction of the two quantifier states \forall and Φ , as can readily be verified by noting that $q(4) = q(1)$ or $q(5)$. The reason why this state is also included in Table 1 is that it is the only additional state in BA-3, obtained by conjunctions and disjunctions of the other seven others listed in it, in order to complete the Boolean algebra BA-3 obeyed by the quantifiers (see MR-52). For this state,

*The symbols \oplus and \otimes for quantifier states employed in (4a,b,c) are really valid in BA-3 algebra, but is put here to stand for the BA-1 operators \vee and \wedge using the truth values of the quantifier states as given by Eq. (1a,b,c), which are equivalent to the union and intersection of the corresponding ranges of m .

.4.

N-2
3.2.87
Draft-2

$q(1)$ is T if a given quantifier state belongs either to $q(1)$, namely "true for all", or $q(4)$, namely "true for none", so that it gives the truth value T for either $q(1)$ or $q(5)$. In either case, the truth value for \ominus is $T \vee F$ or $F \vee T$, both of which are equal to T.

Similarly, for Δ , the symbology $q(1)$ or $q(3)$ or $q(5)$ indicates that a given state can be any one of the possibilities $m = 0$ to N , and will give a truth value T for at least one of $q(1)$, $q(3)$, $q(5)$, so that the r.h.s of (4b) is always T. Null set, on the other hand, requires that a given state must belong to all three generator states $q(1)$ and $q(3)$ or $q(5)$. This is obviously impossible, because the ranges of m for these three states are disjoint, so that, if the state under consideration belongs to $q(1)$, it does not belong to $q(3)$ or $q(5)$ / etc. Consequently, the r.h.s of (4c) is always negative, which means that the range of m is nil, so that the quantifier state is itself a / non-existent one.

The eight quantifier states are permuted among themselves by the application of the operators \underline{E} , \underline{L} , \underline{M} , \underline{N} and no other quantifier states can be generated by combining these by logical connectives. This proves the completeness of QBA. (See the next report N-3.)

.5.

N-2
4.2.87
Draft-22. Derivation of QL-1,2 transformations discussed in N-1

We shall now examine the consequences of the definition given above for a quantifier state, and prove the Equations in (9) and (10) of N-1 using this limited definition of a quantifier. Before that, we shall establish a set of still more general results stated in (5a,b) and (6a,b):

$$f(ax \vee b) = f(ax) \vee b, \quad f(ax \wedge b) = f(ax) \wedge b, \quad (5a,b)$$

$$(f(ax) \quad b) = (f^{\ell}(ax) \quad b)), \quad (f(ax) \quad b) = f^{\ell}(ax \Rightarrow b)) \quad (6a,b)$$

In these, $f(ax)$ is a general logical function of ax , with the locations of \neg, \wedge, \vee , and parantheses $(,)$, defined.

We start with the Eqs.(7a) and (7b) which follow from the ^{distributive} ~~mutually~~ / property of "and", "or":

$$(a_1x \wedge b) \vee (a_2x \wedge b) = (a_1x \vee a_2x) \wedge b \quad (7a)$$

$$(a_1x \wedge b) \wedge (a_2x \wedge b) = (a_1x \wedge a_2x) \wedge b \quad (7b)$$

This is readily generalized to $f(ax \vee b)$ and $f(ax \wedge b)$ of the forms given in (5a,b). The connective between ax and b on the l.h.s is the same as the connective between $f(ax)$ and b in the r.h.s, for \wedge and \vee .

.6.

N-2
5.2.87
Draft-2

In a similar manner, the proof of (6a) and (6b) follows by the steps given in (8a) and (8b). Thus,

$$\begin{aligned} f(ax \Rightarrow b) &= f(\neg ax \vee b) = f(\neg ax) \vee b \\ &= \neg f^b(ax) \vee b = f^b(ax) \quad b \end{aligned} \quad (8a)$$

$$\begin{aligned} (f(ax) \Rightarrow b) &= \neg f(ax) \vee b = f^b(\neg ax) \vee b \\ &= f^b(\neg ax \vee b) = f^b(ax \Rightarrow b) \end{aligned} \quad (8b)$$

If for $f(ax)$ we substitute the appropriate function corresponding to $q(iax)$ in Eq.(1) with the corresponding values of N_1 and N_2 , we get straightaway all the QL-1,2 transformation equations given in the report N-1, involving $f(ax)$ and b .

In a similar way, we can prove the common equation between a and by in (9):

$$f(a Z by) = a Z f(by), \text{ for } Z = \wedge, \vee, \Rightarrow \quad (9)$$

by using the Eqs.(10a,b) corresponding to (7a,b) respectively :

$$(a \wedge b_1y) \vee (a \wedge b_2y) = a \vee (b_1y \wedge b_2y) \quad (10a)$$

$$(a \wedge b_1y) \wedge (a \wedge b_2y) = a \wedge (b_1y \wedge b_2y) \quad (10b)$$

In fact, Eqs.(8) and (9) and the corresponding ones connecting a and by can be given in the form of a pair of

.7.

N-2
4.2.87
Draft-2

equations, as in (11) and (12) — namely*

$$f(\underline{ax}) \underline{Z}(k, \ell) \underline{by} = f'(\underline{ax} \underline{Z}(k, \ell) \underline{by}) ,$$

where

$$f' = f \text{ or } f^\ell \quad \text{according as } k = 1 \text{ or } 2 \quad (11)$$

$$\underline{a} \underline{Z}(k, \ell) f(\underline{by}) = f'(\underline{a} \underline{Z}(k, \ell) \underline{by})$$

where

$$f' \text{ or } f^\ell \quad \text{according as } \ell = 1 \text{ or } 2 \quad (12)$$

The corresponding transformation equations for $q(\underline{iax})$ and $q(\underline{jby})$, $i, j = 1$ to 8 are written by replacing $f(\underline{ax})$ and/or $f(\underline{by})$ by $q(\underline{iax})$ and $q(\underline{jby})$. These two equations can also be combined to give one general transformation equation between QL-1 and QL-2 forms in quantifier logic. This is given in (13a,b).

$$\underline{ax} \ q(\underline{iZx}) \ \underline{Z}(k, \ell) \ q(\underline{jZy}) \ \underline{by} \equiv q(\underline{i'Zx}) \ q(\underline{j'Zy}) \ (\underline{ax} \ \underline{Z}(k, \ell) \ \underline{by}) \quad (13a)$$

where

$$i' = i \text{ or } i^\ell \quad \text{according as } k = 1 \text{ or } 2 , \quad j' = j \text{ or } j^\ell \quad \text{according as } \ell = 1 \text{ or } 2 \quad (13b)$$

and $\underline{Z} = \underline{A} \text{ or } \underline{Q}$.

We shall discuss in the next report some of the symmetry properties of quantifiers which come out of the inter-relationships between the general functions f, f^ℓ, f^m, f^n described here, and follow it up by a description of their exact logical interpretations.

*The extension from \underline{ax} and \underline{by} of BA-1 to \underline{ax} and \underline{by} having BA-2 truth values will be discussed in the next report N-3. However, the generalized formulae in BA-3 are given here.

N-3
11.2.87
Draft-2

Theory of quantifiers employing BA-2 algebra for truth values

1. Extension of Eqs. (1a,b,c) of N-2 to BA-2 truth values

It will be noticed that Eq.(1) of N-2 contains a complete definition in propositional calculus of the logical function representing the four standard quantifiers, as well as the four others that are contained in QBA. It is interesting that exactly similar formulae could be written for BA-2 truth values of SNS also. It is only necessary to replace ax by \underline{ax} , \forall by \underline{Q} , \wedge by \underline{A} and increase m, n to m, n, p , representing the number of terms having the truth values T, F, D. Thus we have

$$\underline{Q}(ax) = \bigcirc_{m=M_1}^{M_2} \bigcirc_{n=N_1}^{N_2} f(\underline{ax}, m, n, p), \quad m + n + p = N \quad (1a)$$

where

$$f(ax, m, n, p) = \bigcirc_{j=1}^{N_{mn}} f_j(\underline{ax}, m, n, p), \quad m + n + p = N \quad (1b)$$

and

$$f_j(ax, m, n, p) = \bigtriangleup_{r=0}^m t(\underline{ax}_r | T) \bigtriangleup_{s=0}^n t(ax_s | F) \bigtriangleup_{t=0}^p t(\underline{ax}_t | D) \quad r + s + t = N \quad (1c)$$

where $t(\underline{ax} | s(k))$ is the truth value of the term \underline{ax} relative to the three permissible states in BA-2, namely $s(1) = T$, $s(2) = F$ and $s(3) = D$ respectively*. The value of the multiple \underline{A} 's for $m = 0$ and for $n = 0$ is T, as a summation with no terms in it/ has a truth value T.

It is also seen that the third "multiple and" in (1c) need not be

*More simply the three truth values are \underline{ax}_r , $\neg \underline{ax}_s$ and T respectively (see Section 4 below).

.2.

N-3
6.2.87
Draft-2

calculated as $\underline{t}(\underline{ax} | D)^*$ is always T for the test state D.

The ranges of m, n, p for the eight standard quantifier states $q(i)$ are listed in Table 1. The data in this table give the representation, in terms of BA-2 calculus, of these states, which have the properties described in MR-52, 53, 54.

A demonstrable logical procedure whereby these states can be discovered for a given input set of data $\underline{ax}(x = 1 \text{ to } N)$ is illustrated by Table 2. In this table we give the consequences of applying the test given in Section 5(b) of MR-57 using BA-1 truth values. If these truth values are applied in BA-2, then all the 16 possible combinations of $\underline{t}_1 = \underline{t}(\underline{ax} | \forall)$, $\underline{t}_2 = \underline{t}(\underline{ax} | \exists)$, produce^{only} the values given in Table 2.

The remaining ones are seen to be self-contradictory and will not be obtained as a result of the calculation from any function $q(\underline{ax})$ of the type in Eq.(1).

However, two of the entries corresponding to $q(4) = \ominus$ and $q(7) = \Delta$ produce the same values (D, D) for $(\underline{t}_1, \underline{t}_2)$. This is because the check is made only for the existence, or non-existence, of the test states \forall and \exists . If it would be possible to devise a logical check whereby the value of $\underline{t}_3 = \underline{t}(\underline{ax} | \Sigma)$

*Definition

$\underline{t}(\underline{a} | q(i)) \equiv (\underline{t}_\alpha \ \underline{t}_\beta)$, where $\underline{t}_\alpha = \langle \underline{a} | \underline{E} | q(i) \rangle$, $\underline{t}_\beta = \langle \underline{a} | \underline{M} | q(i^m) \rangle$
 $\underline{t}(\underline{a} | s(k)) = (\underline{t}_\alpha \ \underline{t}_\beta)$, where $\underline{t}_\alpha = \langle \underline{a} | \underline{E} | s(k) \rangle$, $\underline{t}_\beta = \langle \underline{a} | \underline{M} | s(k^c) \rangle$
 $\underline{t}(\underline{a} | b) \neq \underline{t}(\underline{b} | \underline{a})$ in general (see MR-56, Lecture 4 for details.)

.3.

N-3
6.2.87

Table 1 Theoretical representation of the eight quantifier states in terms of Eqs. (1a,b,c)

i	q(i)	m = M ₁ to M ₂	n = N ₁ to N ₂	p = P ₁ to P ₂	Truth value of q(i) for \forall, Σ, \exists
1	\forall	N N	0 0	0 0	1 0 0
2	\wedge	0 N-1	1 N	0 N-1	0 1 1
3	Σ	1 N-1	1 N-1	2 N-2	0 1 0
4	\ominus	N N	0 0	0 0	1 0 1
	$(\forall \oplus \exists)$	0 0	N N	0 0	
5	$\bar{\exists}$	0 0	N N	0 0	0 0 1
6	\exists	1 N	0 N-1	0 N-1	1 1 0
7	Δ	0 0	0 0	N N	1 1 1
8	ϕ	—	—	—	0 0 0

Table 2. Practical determination of a quantifier state in terms of \forall and \exists

	$\tilde{a}x$	$\tilde{t}(\tilde{a}x A)$	$\tilde{t}(\tilde{a}x E)$	$\tilde{t}(\tilde{a}x \exists)$	$\tilde{t}(\tilde{a}x \Phi)$	$\tilde{t}(\forall a)$	$\tilde{t}(\exists a)$	$\tilde{t}(\Phi a)$
q(1)	\forall	T	T	F	F	T	F	F
q(2)	\exists	F	D	D	D	F	T	T
q(3)	\exists	F	T	T	F	F	T	T
q(4)	\exists^*	D	D*	F	D	T	F	T
q(5)	Φ	F	F	F	T	T	T	F
q(6)	\exists	D	T	D	F	F	T	T
q(7)	∇	D	D	D	D	T	T	T
q(8)	ϕ	X	X	X	X	F	F	F

*These are identical with those for Δ , and is not separately detectable in classical predicate calculus. However, Θ differs from Δ for $\tilde{t}(\tilde{a}x | \exists)$, a parameter only describable in BVMF employing pure and mixed states.

.5.

N-3
9.2.87
Draft-2

can also be determined, then the BA-2 values of t_3 so obtainable are given in the fifth column of the table. It will be then noticed that $q(4)$ and $q(7)$ differ, in that $q(4)$ gives for t_3 the value F, while $q(7)$ yields D. Since the state \sum is not formally available in standard predicate calculus, it seems to be rather difficult to obtain a procedure for checking the existence of \sum as such, in terms of standard predicate calculus. There seems to be definitely a need for the representation of quantifiers by three BA-1 truth values as $q(i) = (q_\gamma \ q_\delta \ q_\epsilon)$ as we have proposed for the BA-3 representation of quantifier states, even with BA-1 truth values for ax . In fact, if $t(q(i)|ax)$ is calculated for the generator states $q(i) = \forall, \sum, \Phi$, then we obtain the data in the last three columns of Table 2. It is then seen that they yield the truth values identical with those obtained using BA-1 truth values shown in the last three columns of Table 1. This shows the identity of the definition of quantifiers in SNS using BA-2 truth values for the terms quantified, and the standard formulation of classical predicate calculus, with two states of truth in BA-1, so that the definitions of a quantifier state given by Eq.(1) is a natural extension, in SNS, of the definition given in N-2 using classical propositional calculus.

.6.

N-3
9.2.87
Draft-2

2. Need for Boolean connectives \underline{U} and \underline{V} in addition to matrix connectives \underline{A} and \underline{O}

As mentioned above, the need for the Boolean sum and Boolean product for component quantifier states becomes particularly apparent on introducing BA-2 truth values, because in SNS we have four states T, F, D, X, with the property $T \oplus F = D$, $T \otimes F = X$, and it is necessary to distinguish between these two Boolean operators \underline{U} (union) and \underline{V} (vidya) and the matrix operators \underline{O} and \underline{A} . The essential difference is in the nature of the two terms whose truth values are combined. The Boolean operators are used for

$$\underline{a}_1 \underline{U} \underline{a}_2 = \underline{a} , \quad \underline{a}_1 \underline{V} \underline{a}_2 = \underline{a} \quad (2a)$$

in which the two data on the l.h.s refer to the same term, while the matrix operators occur for the equations

$$a \underline{A} b \quad \underline{a} \underline{O} \underline{b} = \underline{c} \quad (2b)$$

where the input truth values belong to different terms. In BA-1, product (\otimes) is equivalent to "and" and sum (\oplus) is equivalent to "or", and the distinction between the two pairs is not specifically mentioned, except that, in applications,

.7.

N-3
9.2.87
Draft-2

the states T 'and' F simultaneously applied to the same term lead to contradiction, and concurrently applied to the same term refers to tautology, standing respectively for X and D. On the other hand, the difference between the two cases is brought out by the distinction in the nature of the two types of operators, in that the Boolean operators lead to two equations of the type in (3a,b), which are different from those for the matrix connectives in (4a,b).

Boolean operators

$$\underline{\cup} : a_{1\alpha} \oplus a_{2\alpha} = a_{\alpha} , a_{1\beta} \oplus a_{2\beta} = a_{\beta} \quad (3a)$$

$$\underline{\cap} : a_{1\alpha} \otimes a_{2\alpha} = a_{\alpha} , a_{1\beta} \otimes a_{2\beta} = a_{\beta} \quad (3b)$$

Matrix operators

$$\underline{\cup} : a_{\alpha} \oplus b_{\alpha} = c_{\alpha} , a_{\beta} \otimes b_{\beta} = c_{\beta} \quad (4a)$$

$$\underline{\cap} : a_{\alpha} \otimes b_{\alpha} = c_{\alpha} , a_{\beta} \oplus b_{\beta} = c_{\beta} \quad (4b)$$

However, as may be seen from the above two pairs of equations, the same BA-1 operators \oplus and \otimes are employed in both cases. This distinction between ^{and} "and" / "or" on the one hand, and ^{and} "union" / ^{other,} "vidya" on the ~~the~~ is not there in standard Boolean algebra, so that there is no complete isomorphism between the algebra of logic and Boolean algebra in its standard form. However,

.8.

N-5
9.2.87
Draft-2

all the formulae can be reduced to those involving only the BA-1 operators of complementation (c), sum (\oplus) and product (\otimes), which have been adopted for what we have called matrix operators in classical logic. The distinction between A, O and U, V in BA-1 itself does not seem to have been recognized as an algebraic fact/and that it is capable of being represented/only in BA-2, although contradiction and tautology are concepts which are fundamental and well understood in that field.

3. Boolean algebraic representation of connectives in predicate logic

Therefore, we shall consider in some detail the nature of the correct connective operators in BA-3 which correspond to Eqs. (3) and (4) in BA-2 for quantifier logic in general. On ~~careful~~ ^{detailed} examination, it turns out that we have 3×3 matrix operators similar to 2×2 matrix connectives of BA-2. As regards Boolean connectives, four different operators $\underline{E}, \underline{N}, \underline{M}, \underline{L}$ are required. Of these, \underline{E} and \underline{N} are matrices having all their non-zero elements equal to 1 with the BA-1 connotation of equivalence (e), and \underline{M} and \underline{L} have complementation (c) as the elements corresponding to 1, in the two diagonals of the

.9.

N-3
9.2.87
Draft-2

matrices $|E|$ and $|N|$ respectively. Because of this, we shall call these as Boolean-matrix operators. For ready reference, both the Boolean equations corresponding to them, as well as their matrix representation, are given in Eqs. (5) and (6).

Boolean representation

$$\begin{aligned}
 \underline{E} &: a_\gamma = b_\gamma, \quad a_\delta = b_\delta, \quad a_\epsilon = b_\epsilon \\
 \underline{N} &: a_\gamma = b_\epsilon, \quad a_\delta = b_\delta, \quad a_\epsilon = b_\gamma \\
 \underline{M} &: a_\gamma^c = a_\gamma, \quad a_\delta^c = b_\delta, \quad a_\epsilon^c = b_\epsilon \\
 \underline{L} &: a_\gamma^c = b_\epsilon, \quad a_\delta^c = b_\delta, \quad a_\epsilon^c = b_\epsilon
 \end{aligned} \tag{5}$$

Matrices

$$\begin{aligned}
 |E| &= \begin{pmatrix} e & 0 & 0 \\ 0 & e & 0 \\ 0 & 0 & e \end{pmatrix}, & |N| &= \begin{pmatrix} 0 & 0 & e \\ 0 & e & 0 \\ e & 0 & 0 \end{pmatrix} \\
 |M| &= \begin{pmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{pmatrix}, & |L| &= \begin{pmatrix} 0 & 0 & c \\ 0 & c & 0 \\ c & 0 & 0 \end{pmatrix}
 \end{aligned} \tag{6}$$

We shall not discuss 3x3 matrix connectives of quantifier logic, in general, since this has been done in MR-52, 53, 54, However, we shall consider some consequences regarding the symmetry of the BA-3 Boolean-matrix connectives and the non-equivalence of standard/Boolean algebra and the Boolean algebraic

representation of logic. It turns out that unless some symbols are used to represent tautology and contradiction and also the distinction between A, O and U, V, it is not possible to write algorithmic formulae which can be implemented by a computer. This is considered in the next section.

4. Non-distributivity of logical functions

We shall illustrate this by a very simple example of a formula written in classical logic, namely

$$f = a \wedge (\neg a \vee b) \equiv (a \wedge \neg a) \vee (a \wedge b) \quad (7a,b)$$

With the substitutions given below, both the l.h.s and the r.h.s of (7) can be put in the form of elementary relations as given in (8) and (9).

$$\text{l.h.s of (7) : } \neg a \vee b = g, \quad a \wedge g = f_1 \quad (8a,b)$$

$$\text{r.h.s of (7) : } a \wedge \neg a = h, \quad a \wedge b = k, \quad h \vee k = f_2 \quad (9a,b,c)$$

Then, with $\underline{a} = T, \underline{b} = T$, we have

$$\underline{g} = T, \quad \underline{f_1} = T \quad (a) ; \quad \underline{h} = X, \quad \underline{k} = T, \quad \underline{f_2} = X \quad (b) \quad (10)$$

It is seen that the two expressions do not lead to the same truth value for \underline{f} , $\underline{a} = T, \underline{b} = T$, which is what is meant by the symbols \underline{a} and \underline{b} . We have used the SNS notation of X to indicate what ,

in classical logic would be called contradiction. The point of issue is that the expressions (7a) and (7b) are not logically equivalent, although, on removing the brackets and regrouping the terms, one is equivalent to the other in BA-1 by substituting \oplus for \vee and \otimes for \wedge . Thus, although it is obvious that $a \wedge \neg a$ is contradiction (X) and $a \vee \neg a$ is tautology (D), in evaluating the truth value of a expression for given inputs, this fact is incapable of being introduced in Boolean algebra (BA-1) for automatic evaluation. Even if we give the instruction that the connective applied between the same entity (in this case, in $a \wedge \neg a$) is to be treated differently from that between two different terms (as in $a \wedge b$), unless the difference in the truth tables of the two forms of \wedge , and \vee is specifically fed in, it is not possible to implement them automatically. This is, however, equivalent to the BA-2 representation, because BA-2 employs the 2×2 truth table in toto as the ^{Boolean} elements of the 2×2 matrix appropriate to the connective concerned. (See Lecture 1, Series 3 for a ^{fuller exposition}).

This example illustrates our main contention that, when a logical graph is implemented, the corresponding algebraic

.12.

N-3
11.2.87
Draft-1

equations should be formulated with the parantheses grouping them together written in the same pattern as the steps in the logical argument, and implemented sequentially in that form, without removing the parantheses and regrouping them before implementation. Therefore, the process of converting a logical formula into the conjunctive normal form, or the disjunctive normal form, is not a valid process in the BVMF_{etc} implementation of logic.

This can be illustrated even more clearly by the definition of a quantifier state $\underline{q}(ax)$ in Eq.(1) above, and particularly in its application to the steps (1b) and (1c). In these, the disjunction of $f_j(\underline{ax}, m, n, p)$ over j in (1b) is to be performed only after the conjunctions in (1c) to obtain this function is performed, and the resultant expression cannot be distributed. For instance, for $N = 2$, consider the function $\underline{q}(ax)$ in Eq.(1) to be \exists . Then, by Table 1, at least one \underline{ax} must be T, as a consequence of which only the three combinations (1, 0, 1), (1, 1, 0), (2, 0, 0) are permitted for m, n, p . Therefore, the expression for $\underline{q}(ax) = \exists$ takes the form of

.13.

N-3
12.2.87
Draft-2

Eqs (11) and (12) in the same format as Eqs. (1a,b,c):

$$\underline{t}(q(\underline{ax}) \mid \exists) = (f(\underline{ax}, 1, 0, 1)) \underline{\underline{O}} (f(\underline{ax}, 1, 1, 0)) \underline{\underline{O}} (f(\underline{ax}, 2, 0, 0)) \quad (11)$$

where, omitting \underline{ax} for convenience, we have

$$f(1, 0, 1) = \underline{t}(\underline{a_1} \mid T) \underline{\underline{O}} \underline{t}(\underline{a_2} \mid T) \quad (12a)$$

$$f(1, 1, 0) = (\underline{t}(\underline{a_1} \mid T)) \underline{\underline{A}} (\underline{t}(\underline{a_2} \mid F)) \underline{\underline{O}} (\underline{t}(\underline{a_2} \mid T) \underline{\underline{A}} \underline{t}(\underline{a_1} \mid F)) \quad (12b)$$

$$f(2, 0, 0) = \underline{t}(\underline{a_1} \mid T) \underline{\underline{A}} \underline{t}(\underline{a_2} \mid T) \quad (12c)$$

The extra brackets in $(f(\underline{ax}, m, n, p))$ is used to indicate that (12a,b,c) have to be separately calculated first, and then fed in the r.h.s of (11).

It will be seen that, for the sample $\underline{a_1} = T, \underline{a_2} = F$,

$$f(1, 0, 1) = \underline{t}(T \mid T) \underline{\underline{O}} \underline{t}(F \mid T) = T \underline{\underline{O}} F = T \quad (13a)$$

$$f(1, 1, 0) = (T \underline{\underline{A}} T) \underline{\underline{O}} (F \underline{\underline{A}} F) = T \underline{\underline{O}} F = T \quad (13b)$$

$$f(2, 0, 0) = T \underline{\underline{A}} F = F \quad (13c)$$

Substituting these in Eq.(11), we obtain

$$\underline{t}(q(ax) \mid \exists) = T \underline{\underline{O}} T \underline{\underline{O}} F = T \quad (14)$$

showing that the quantifier state to which the sample set

$(\underline{a_1} = T, \underline{a_2} = F)$ belongs is $(\exists x)(\underline{ax})$. It will be noticed that

only one $f_j(\underline{ax}, r, s, t)$ namely $f(2, 0, 0)$ has the truth

value F for the sample input, while the other two lead to T .

.14.

N-3
12.2.87
Draft-2

On the other hand, this sample ($\underline{a}_1 = T, \underline{a}_2 = F$) does not belong to $(\forall x)(ax)$, since Eq.(1) leads to Eq.(15) for the only value of $f_j(\underline{ax}, 2, 0, 0)$ shown therein.

$$t(\underline{q}(\underline{ax}) | \forall) = f(ax, 2, 0, 0) = F \quad (15)$$

However, the sample set ($\underline{a}_1 = T, \underline{a}_2 = T$) belongs to the quantifier state \forall , since

$$\underline{t}(\underline{q}(\underline{ax}) | \forall) = f(2, 0, 0) \text{ of (12c)} = T \wedge T = T \quad (16)$$

This is a very simple and trivial example. However, using the formula (13b), we shall illustrate the non-distributivity of \wedge and \vee in it if only BA-1 is employed, and the need to put the brackets where they are. Writing this equation in BA-1, but retaining the BA-2 symbols T, F, D, X for these states, Eq.(12b) takes the form of (17) since $\underline{t}(a_i | T) = a_i$, $\underline{t}(a_i | F) = \neg a_i$ in classical logic.

$$(a_1 \wedge \neg a_2) \vee (\neg a_1 \wedge a_2) = f \quad (17)$$

Its value according to (13b) is

$$(T \wedge T) \vee (F \wedge F) = T \vee F = T = f_1 \quad (18)$$

If, however, the expression (17) is expanded, we have

$$(a_1 \vee \neg a_1) \wedge (\neg a_2 \vee \neg a_1) \wedge (a_1 \vee a_2) \wedge (\neg a_2 \vee a_2) = f \quad (19)$$

.15.

N-3
12.2.87
Draft-2

In this, it will be noticed that the two terms $(a_1 \vee \neg a_1)$ and $(\neg a_2 \vee a_2)$ are both tautologies in classical logic, and can be replaced by $D (= T \oplus F \text{ in BA-2})$. For the middle two terms, the truth values can be calculated by using the logical connective \vee as being equivalent to $\underline{0}$ of BA-2. Then we have

$$D \wedge (T \vee F) \wedge (T \vee F) \wedge D = D \underline{A} T \underline{A} T \underline{A} D = D = f_2 \quad (20)$$

It is clear that the value $f_1 = T$ in (18) is different from the value $f_2 = D$ in (20), although the two expressions (17) and (19) are equivalent in BA-1. This arises because the symbol \vee has two different connotations — namely of \underline{U} (/ "sum") ^{Boolean operator} in the first and last terms of (19), and that of $\underline{0}$ (/ "or") ^{matrix operator} in the middle two terms. Obviously, what is required in the definition of the quantifier state is the value T for f as in (17), with brackets as given in (12b), and this becomes D if the brackets are removed and the expression expanded as in (19).

This shows the non-distributive nature of \underline{U} , or \vee , against the corresponding logical connectives $\underline{0}$, or \wedge , and the need to keep the parantheses intact if the formula of a compound logical term has been translated into BVMF notation. In the Boolean algebraic manipulations, no expansion leading to the removal of

.16.

N-3
12.2.87
Draft-2

parantheses should be carried out. With this precaution, BVMF is a faithful representation in BA-2 of any formula in logic and the equation so specified involving conjunctions, disjunctions, and negations, is unique and can be implemented in a unique manner. There is no ambiguity in the meaning of \vee and \wedge , but its nature can be specified to be either $\underline{\mathbb{A}}$ or $\underline{\mathbb{V}}$, or $\underline{\mathbb{O}}$ or $\underline{\mathbb{U}}$. However, at the same time, we lose the possibilities of converting any general logical polynomial into its conjunctive or disjunctive normal form. This is, however, of no consequence, since the BVMF technique is one of implementing relations one at a time, and therefore, the logical graph is always constructible and implementable in the form in which the argument is given. In fact, the considerations given above do not affect any of the arguments in MR-57 or the previous notes N-1 and N-2.

5. Symmetry properties of quantifier states and their connectives

In this section we shall mention some interesting symmetry properties between the quantifier states as defined by Eq.(1) and whose properties are specified by Table 1. The discussion is common to the standard definition via BA-1 algebra and the BVMF definition via the four possible states T, F, D, X in BA-2.

In N-1, we have shown that a general logical function

$f(a_1)$ in propositional calculus has four modifications

$$f(a_1), f^{\ell}(a_1) = f^*(a_1), f^m(a_1) = \neg f(a_1), f^n(a_1) = f(\neg a_1) \\ = \neg f^*(a_1) \quad (21)$$

and that the quantifier states $\forall, \exists, \wedge, \Phi$ have the same relationships via similar operators, leading to the 1-1 corresponden

$$f \leftrightarrow \forall = q(1^e) = q(1), f^{\ell} \leftrightarrow \exists = q(1^{\ell}) = q(6); \\ f^m \leftrightarrow \wedge = q(1^m) = q(2), f^n \leftrightarrow \Phi = q(1^n) = q(5) \quad (22)$$

Because the transformation properties for the logical functions in (21) are valid for all functions in propositional calculus and since a quantifier can always be expressed as a function of logical terms, each having BA-1 or BA-2 truth values as shown in this report, the relationships in (22) can be generalized to cover the case when $f(ax)$ is any one of the standard four quantifier states $\forall, \exists, \wedge, \Phi$. These are best shown in the form of Table 3.

It is interesting that the four SNS states T, F, D, X also have the same transformation properties as the four standard quantifier states $\forall, \exists, \wedge, \Phi$, and are permuted among themselves in a similar manner by the application of the four 2x2 Boolean-matrix operators $\underline{E}, \underline{L}, \underline{M}, \underline{N}$ analogous to the 3x3 Boolean-matrix operators in BA-3. This is shown in Table 4.

Table 3. Transformations of quantifier states by the four Boolean-matrix operators \underline{E} , \underline{L} , \underline{M} , \underline{N}

	(a) Standard quantifier states				(b) New states produced in BA-3			
	\forall		\wedge	Φ	Σ	Δ	ϕ	
\underline{E}	\forall	\exists	\wedge	Φ	Σ	Δ	ϕ	
	\exists	\forall	Φ	\wedge	Θ	Σ	ϕ	Δ
\underline{M}	\wedge	Φ	\forall	\exists	Σ	ϕ	Δ	
\underline{N}	Φ	\wedge	\exists	\forall	Σ	Δ	ϕ	

Table 4. Transformations of the SNS states T, F, D, X by the four Boolean-matrix operators \underline{E} \underline{L} \underline{M} \underline{N}

	Classical states		Additional states in SNS	
	T	F	D	X
\underline{E}	T	F	D	X
\underline{N}	F	T	D	X
\underline{M}	F	T	X	D
\underline{L}	T	F	X	D

.19.

N-3
Draft-2
16.2.87

Table 3,

However, in the two pairs of states $(\underline{\Sigma}, \underline{\Theta})$ and $(\underline{\Delta}, \underline{\phi})$ are only converted one into the other as a result of the application of all the four operators. In fact, $\underline{\Sigma} \underline{N} = \underline{\Sigma}$ itself and $\underline{\Sigma} \underline{M} = \underline{\Theta}$. The former relation is indicative of the fact that $(\underline{\Sigma} x)(\underline{a}x) \equiv (\underline{\Sigma} x)(\neg \underline{a}x)$. The same property is held by $\underline{\Theta}$, $\underline{\Delta}$ and $\underline{\phi}$, all of which are symmetric in (ax) and $(\neg ax)$. Just as $\underline{\Sigma} \underline{M} = \underline{\Theta}$ and $\underline{\Theta} \underline{M} = \underline{\Sigma}$, $\underline{\Delta}$ and $\underline{\phi}$ are also complements of one another in the form $\underline{\Delta} \underline{M} = \underline{\phi}$ and $\underline{\phi} \underline{M} = \underline{\Delta}$. The consequences for the implementation of various relations in predicate calculus have been treated in MR-53 and 54.

The four operators \underline{E} , \underline{L} , \underline{M} , \underline{N} form a group of order 4, the so-called Vierergruppe (V), which is isomorphic with the crystallographic point group $D_2 \equiv 2 \ 2 \ 2$. The latter notation is indicative of the fact that there are three elements in the group of order 2 (in our case, $\underline{M}^2 = \underline{N}^2 = \underline{L}^2 = \underline{E}$), although the group is generated by any two of these. Taking these to be \underline{M} and \underline{N} , $\underline{L} = \underline{M} \underline{N} = \underline{N} \underline{M}$, and consequently, $\underline{L} \underline{M} = \underline{M} \underline{L} = \underline{N}$, and $\underline{L} \underline{N} = \underline{N} \underline{L} = \underline{M}$ also follow for this group. We shall not discuss these features as they are only of theoretical interest, but consider essentially the logical consequences of this formalism.

The most interesting consequence of this study of the relationships between the quantifier states of QBA is that we have arrived at two types of negations for these states, namely \underline{N} (negation) and \underline{M} (complementation). The former negates the predicate $\underline{a}x$ in the expression $(\underline{q}x)(\underline{a}x)$, while the latter negates the quantifier state $(\underline{q}x)$ and not the predicate. Therefore, it is highly desirable to have two symbols for these two types of negation — "weak negation" $\underline{N}(\neg)$ and "strong negation" $\underline{M}(\neg)$. The words weak and strong had been coined

.20.

N-3
Draft-1
13.2.87

for similar operations which can be postulated for propositional calculus also, which will be described below, and the reason for the terminology is that, while weak negation \underline{N} makes T go into F, and F into T, leaving D and X unchanged, the strong negation \underline{M} makes all four of them go into their complementary states. Therefore, in all our formulae, we shall use these two symbols \neg and \neg to distinguish between \underline{N} and \underline{M} .

Table 4, which is closely similar in pattern to Table 3, gives the transformation of the four SNS states produced by four operators analogous to those discussed above, namely \underline{E} , \underline{L} , \underline{M} , \underline{N} . The sequence of the operators is made slightly different from Table 3 for reasons that will be clear below.

The Boolean representation of the relation $\underline{a} \underline{R} = \underline{b}$, between $(a_\alpha \ a_\beta)$ and $(b_\alpha \ b_\beta)$, and the corresponding matrix $|R|$, are given in Eqs. (23) and (24) in the same pattern as Eqs. (5) and (6) above.

Boolean representation

$$\begin{aligned} \underline{E} : \quad a_\alpha &= b_\alpha, \quad a_\beta = b_\beta; \quad \underline{N} = \quad a_\alpha = b_\beta, \quad a_\beta = b_\alpha; \\ \underline{M} : \quad a_\alpha^c &= b_\alpha, \quad a_\beta^c = b_\beta; \quad \underline{L} = \quad a_\alpha^c = b_\beta, \quad a_\beta^c = b_\alpha \end{aligned} \quad (23)$$

.21.

N-3
Draft-2
13.2.87Matrices

$$\begin{aligned}
 |E| &= \begin{pmatrix} e & 0 \\ 0 & e \end{pmatrix}, & |N| &= \begin{pmatrix} 0 & e \\ e & 0 \end{pmatrix} \\
 |M| &= \begin{pmatrix} c & 0 \\ 0 & c \end{pmatrix}, & |L| &= \begin{pmatrix} 0 & c \\ c & 0 \end{pmatrix}
 \end{aligned} \tag{24}$$

Of the four matrices, the first two, namely $|E|$ and $|N|$, have been incorporated in our SNS formalism in the forms $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ respectively which are equivalent to the Boolean-matrix form given in (24). \underline{M} , on the other hand, had been written in the Boolean representation as given in (23), but we now notice that it can also be put in the Boolean-matrix form as in (24). The operator \underline{L} has not been found to be of great use for logical relations in propositional calculus, except in the case of functions $f(\underline{a}_i)$, which lead to equations of the type (21) which are highly useful for the definition of quantifiers.

The most interesting application of the unification, of the formalism for Boolean operators in BA-2 and BA-3, is the possibility of a clarified presentation of the De Morgan relations connected with the operators \underline{A} , \underline{O} on the one hand, and \underline{V} , \underline{U} on the other, in SNS algebra. The essentially new

.22.

result is, that the weak negation $\underline{\underline{N}}$ is what is operative between $\underline{\underline{A}}$ and $\underline{\underline{O}}$, and the strong negation $\underline{\underline{M}}$ is the one that is operative between $\underline{\underline{U}}$ and $\underline{\underline{V}}$, as given in Eqs. (25) and (26) below.

$$\neg \underline{\underline{a}} \underline{\underline{A}} \neg \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{N}} \underline{\underline{A}} \underline{\underline{N}} \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{O}}^c \underline{\underline{b}} = \neg (\underline{\underline{a}} \underline{\underline{O}} \underline{\underline{b}}) \quad (25a)$$

$$\neg \underline{\underline{a}} \underline{\underline{O}} \neg \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{N}} \underline{\underline{O}} \underline{\underline{N}} \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{A}}^c \underline{\underline{b}} = \neg (\underline{\underline{a}} \underline{\underline{A}} \underline{\underline{b}}) \quad (25b)$$

These are the equivalents of the standard De Morgan relation in BA-1, employing the operators $\otimes, \oplus, ^c$, for \wedge, \vee, \neg for Boolean scalars in BA-1. These three operators $\otimes, \oplus, ^c$, go over into the SNS operators $\underline{\underline{N}} (\neg), \underline{\underline{A}} (\wedge), \underline{\underline{O}} (\vee)$, in BA-2 for this logic. On the other hand, $\underline{\underline{V}}$ and $\underline{\underline{U}}$ are formally two new operations introduced in BA-2 and they satisfy the De Morgan relations (26a,b).

$$\neg \underline{\underline{a}} \underline{\underline{V}} \neg \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{M}} \underline{\underline{V}} \underline{\underline{M}} \underline{\underline{b}} = \neg (\underline{\underline{a}} \underline{\underline{U}} \underline{\underline{b}}) \quad (26a)$$

$$\neg \underline{\underline{a}} \underline{\underline{U}} \neg \underline{\underline{b}} = \underline{\underline{a}} \underline{\underline{M}} \underline{\underline{U}} \underline{\underline{M}} \underline{\underline{b}} = \neg (\underline{\underline{a}} \underline{\underline{V}} \underline{\underline{b}}) \quad (26b)$$

In this case, the three operations \otimes, \oplus and c of BA-1 go into $\underline{\underline{M}} (\neg), \underline{\underline{V}} (\otimes)$ and $\underline{\underline{U}} (\oplus)$.

It is obvious that the two operations $\underline{\underline{M}}$ and $\underline{\underline{N}}$ are equivalent for the classical states T and F, since they go over into F and T

respectively, for both \underline{N} and \underline{M} , although they differ in that \underline{D} goes over to X and X goes over to D for \underline{M} , while they remain unchanged by the operation of \underline{N} . Therefore, the distinction between these two will not be detectable in classical logic employing only two states T and F , for which both (25a,b) and (26a,b) lead to Eqs. (27a,b) given below.

We shall now prove (25a) and (26a) to indicate how this distinction between the \neg and \neg occurs in BA-2. Both the equations can be proved from the primitive De Morgan relations in BA-1, as in (27a,b) below.

$$a^c \otimes b^c = (a \oplus b)^c ; \quad a^c \oplus b^c = (a \otimes b)^c \quad (27a,b)$$

The difference between (25) and (26) arises because the two equations between the components $(a_\alpha \ a_\beta)$ and $(b_\alpha \ b_\beta)$ are different for \underline{A} and for \underline{V} , as shown in Eqs. (3) and (4).

As a consequence of this, the two operators N and M are differently invoked in the two cases when the common BA-1 De Morgan relations (27) are extended to BA-2. Thus, considering (25a), we have,

$$\begin{aligned} \neg \underline{a} \underline{A} \neg \underline{b} &\equiv (a_\alpha \ a_\beta) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} |A| \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} b_\alpha \\ b_\beta \end{pmatrix} \\ &= \langle (a_\beta \ a_\alpha) |A| (b_\beta \ b_\alpha) \rangle \\ &= (a_\beta \otimes b_\beta \ a_\alpha \oplus b_\alpha) = (a_\alpha \oplus b_\alpha \ a_\beta \otimes b_\beta) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \neg (d_\alpha \ d_\beta) \end{aligned} \quad (28a)$$

where

$$\underline{a} \underline{O} \underline{b}$$

(28b)

In the same way, if we consider (26a) for the Boolean connective $\underline{\vee}$, we have

$$\begin{aligned}\neg \underline{a} \underline{\vee} \neg \underline{b} &= (a_{\alpha}^c \ a_{\beta}^c) \otimes (b_{\alpha}^c \ b_{\beta}^c) \\ &= (a_{\alpha}^c \otimes b_{\alpha}^c \ a_{\beta}^c \otimes b_{\beta}^c) = ((a_{\alpha} \oplus b_{\alpha})^c \ (a_{\beta} \oplus b_{\beta})^c) \\ &= \neg (c_{\alpha} \ c_{\beta})\end{aligned}\quad (29a)$$

$$\text{where} \quad \underline{c} = \underline{a} \underline{\cup} \underline{b} \quad (29b)$$

As already mentioned, the two sets of equations in (28) and (29) are equivalent for the classical truth values T and F for a term. The difference is noticeable only for D and X, where the matrix operator \underline{N} interchanges α and β components, while the matrix operator \underline{M} complements, in BA-1, the two components a_{α} and a_{β} of a term \underline{a} . Both these negation operators \underline{M} and \underline{N} , when extended to BA-3 as \underline{M} and \underline{N} , are applicable in predicate logic.

It is interesting that $\neg \underline{a} \underline{A} \neg \underline{b}$ and $\neg \underline{a} \underline{O} \neg \underline{b}$ do not have simple De Morgan type equivalents, as \neg is a Boolean operator and \underline{A} and \underline{O} are matrix operators. However, for the pair \underline{U} , \underline{V} , there are such equations for all four Boolean-matrix type operators \underline{E} , \underline{N} , \underline{M} , \underline{L} as in (30a,b) below:

$$\underline{a}^e \underline{V} \underline{b}^e = \underline{c}^e, \ \underline{a}^n \underline{V} \underline{b}^n = \underline{c}^n; \quad \text{where} \quad \underline{c} = \underline{a} \underline{V} \underline{b} \quad (30a)$$

$$\underline{a}^m \underline{V} \underline{b}^m = \underline{d}^m, \ \underline{a}^{\ell} \underline{V} \underline{b}^{\ell} = \underline{d}^{\ell}; \quad \text{where} \quad \underline{d} = \underline{a} \underline{U} \underline{b} \quad (30b)$$

The corresponding relation for \underline{U} on the l.h.s is obtained by replacing \underline{U} , \underline{V} in (30a,b) by \underline{V} , \underline{U} . The proof of all these follow by the same argument as in (29).

The consequences of the two types of negations \underline{M} and \underline{N} and the consequent existence of the operator \underline{L} , which we shall call as "parallel", will be discussed in a later report.

Tensor Algebra in BVMF

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Tensor Algebra in BVMF

G.N. Ramachandran

INSA Albert Einstein Professor
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

Matphil Report No. 59, May 1987.

PREFACE

The material presented here was written soon after MR-57 was completed, and is a treatment of the generalization of the tensor calculus of ordinary algebra (as used in crystal physics or general relativity) to Boolean algebra. The formulae have all been obtained by analogy, and require no proof as they are mostly in the form of definitions. Thus, the analogues of "Contracted" sums, and "direct" (expanded) sums are the corresponding Boolean "sums" or "products", and in addition we can also have "simple" sums and products in Boolean algebra. All these have applications to the BVMF representation of logic.

These ideas have found very good application in the development of computer-implementable algorithms for predicate logic, presented in MR-60,61,62.

Essential mathematical formulae for BVMF in general

As the name Boolean Vector-Matrix Formalism indicates, the entities dealt with are Boolean scalars, Boolean m-vectors, Boolean matrices (m x n) and higher order Boolean tensors. These have descriptions as below in Eqs. (1) to (4)

1. Boolean entities and BA-1 algebra required for the theory

Scalar (0 order tensor) c having the Boolean values 1, 0 . The quantities $a_1, R_{ij}, R_{i1}, i_2, \dots, i_r$ ^{below} are all Boolean numbers having only two values 1, 0 . (1)

Vectors : $\underline{a} = (a_1, a_2, \dots, a_m), \equiv \{a_i\}, i = 1 \text{ to } m$ (2)
(1st order tensor)

Matrix (2nd order tensor) :

$$\underline{R} = \{R_{ij}\}, i = 1 \text{ to } m, j = 1 \text{ to } n \quad (3)$$

Tensor (higher order = r):

$$\underline{R} = \{R_{i1, i2, \dots, ir}\}, \quad (4)$$

$i1 = 1 \text{ to } n1, i2 = 1 \text{ to } n2, \dots, ir = 1 \text{ to } nr$

Just as standard tensor calculus over the field of real numbers is defined using the standard arithmetical operations of addition and multiplication, the Boolean operations required for BVMF are only three in number, namely

.2.

Boolean sum (\oplus), Boolean product (\otimes) and Boolean complement (c) which are defined in BA-1. We shall give below the analogues of various tensor operations such as scalar product, direct product and contraction.

2. Scalar products and sums, in BVMF

(a) Scalar products

Analogous to the scalar product of two m-vectors, namely

$$\sum_i a_i b_i = c \quad (5a)$$

we have the scalar product of two Boolean m-vectors:

$$\bigoplus_i a_i \otimes b_i = c \quad (5b)$$

Just as we can have ^a/scalar arising out of a second order tensor of the type

$$\sum_i \sum_j R_{ij} a_i b_j = c \quad (6a)$$

we have the scalar triple product (~~Dirac~~ product of two vectors and a matrix) namely,

$$\bigoplus_i \bigoplus_j (a_i \otimes R_{ij} \otimes b_j) = c \quad (6b)$$

and

$$\bigoplus_{i1} \bigoplus_{i2} \dots \bigoplus_{ir} (R_{i1, i2, \dots, ir} \otimes a_{i1} \otimes b_{i2} \otimes \dots \otimes g_{ir}) \quad (7)$$

Unlike in standard vector-tensor analysis, in Boolean algebra, the operation of sum is as important as operation of product, and therefore (5b), (6b) and (7) have analogous equations involving also the Boolean operator "sum". These however, lead to a wide variety of forms, if both sum and product are included and the complement is also included. However, they can all be reduced to a small number of standard forms using the De Morgan relations as will be indicated below:

(b) Analogous operations involving "scalar sums"

In MR-57, more general types of Boolean operations than those considered above in Section (a) have been envisaged and were found to be necessary. Generalizing these still further, we can formulate the following possible combinations for the scalar product and the scalar sum of two m-vectors as in (8), including the one in (5b). They are

$$\bigoplus_i (a_i \otimes b_i) = c \quad (8a)$$

$$\bigotimes_i (a_i \otimes b_i) = c \quad (8b)$$

$$\bigoplus_i (a_i \oplus b_i) = c \quad (8c)$$

$$\bigotimes_i (a_i \oplus b_i) = c \quad (8d)$$

Some of these are interconvertible — e.g.

$$\bigotimes_i (a_i \oplus b_i) = \left(\bigoplus_i (a_i^c \otimes b_i^c) \right)^c \quad (8e)$$

Similarly, analogous to Eq.(6), several possible combinations of the Boolean operations sum and product are available/^{of} which four have been utilized in MR-57 (see Table 3, page 27 and the formulae leading to them). These four are summarized in Eqs.9(a-d).

$$\bigoplus_i \bigoplus_j a_i \otimes R_{ij} \otimes b_j = c \quad (9a)$$

$$\bigoplus_i \bigotimes_j a_i \otimes R_{ij} \otimes b_j = c \quad (9b)$$

$$\bigotimes_i \bigoplus_j a_i \otimes R_{ij} \otimes b_j = c \quad (9c)$$

$$\bigotimes_i \bigotimes_j a_i \otimes R_{ij} \otimes b_j = c \quad (9d)$$

All other possibilities are obtained by substituting for one or both of the two Boolean products in $a_i \otimes R_{ij} \otimes b_j$, the Boolean sum \bigoplus in the four functions occurring in 9(a-d).

In such cases, ^{sequence and} proper/parantheses must be included in the definition of the function on the l.h.s. ^{the expressions} e.g. / (10a) and (10b) are not equivalent.

$$(\bigoplus_i a_i \otimes R_{ij}) \oplus b_j \neq \bigoplus_i (\bigotimes_j R_{ij} \oplus b_j) \otimes a_i \quad (\text{e.g. } 10a, b)$$

All these have logical representations as we shall indicate below and therefore in a generalized treatment of logic as well as Boolean algebra via BVMF representation, these operations have to be taken into account. In fact, (10b) becomes (10c)

by the application of De Morgan relation, which is of the form (10a)

$$\left(\bigoplus_i \left(\bigoplus_j R_{ij}^c \otimes b_j^c \right) \oplus a_i^c \right) \quad (10c)$$

Following the same lines, a general tensor relation as in (7) can also be of a form in which any one of the multiple sums or Boolean products in (7) could be replaced by its parallel type of operator, namely \otimes or \oplus .

In these, we have not utilized the complementation operator in the general formulae, but only in the application of De Morgan relations to get equivalent modified forms. This is in the spirit of standard algebraic practice — e.g. in the formula $\sum_j R_{ij} b_j = a_i$ where a_i and b_j are not explicitly given a negative sign although this may be required in particular examples. In the same way, any one of a_i , b_j , a_{i1} etc., can also have the complementation operator attached to it and the inclusion of this feature is only a matter of detail.

3. Boolean products and sums of two vectors, matrices or tensors

These are generalization of the Boolean sum and Boolean product defined for SNS states, quantifier states, and relations in SNS logic and quantifier logic involving 2x2 and 3x3 Boolean matrices, which have been considered in earlier reports. We shall generalize these and define the standard types of these operations.

Boolean sum of two m -vectors, $\underline{a} \oplus \underline{b} = \underline{c}$:

$$a_i \vee b_i = c_i, \quad i = 1 \text{ to } m \quad (11a)$$

Boolean product of two m -vectors, $\underline{a} \otimes \underline{b} = \underline{c}$

$$a_i \otimes b_i = c_i, \quad i = 1 \text{ to } m \quad (11b)$$

Boolean sum of two $m \times n$ matrices, $\underline{A} \oplus \underline{B} = \underline{C}$

$$A_{ij} \oplus B_{ij} = C_{ij}, \quad i = 1 \text{ to } m, j = 1 \text{ to } n \quad (12a)$$

Boolean product of two $m \times n$ matrices, $\underline{A} \otimes \underline{B} = \underline{C}$

$$A_{ij} \otimes B_{ij} = C_{ij}, \quad i = 1 \text{ to } m, j = 1 \text{ to } n \quad (12b)$$

Boolean sum of two general tensors, $\underline{A} \oplus \underline{B} = \underline{C}$

$$a_{i_1, i_2, \dots, i_r} \vee b_{i_1, i_2, \dots, i_r} = c_{i_1, i_2, \dots, i_r} \quad (13a)$$

Boolean product of two general tensors, $\underline{A} \otimes \underline{B} = \underline{C}$

$$a_{i_1, i_2, \dots, i_r} \otimes b_{i_1, i_2, \dots, i_r} = c_{i_1, i_2, \dots, i_r} \quad (13b)$$

Obviously, in all these cases, one can also have the complementation operator, leading to

$$\bar{a}_i = \bar{b}_i; \quad \bar{A}_{ij} = \bar{B}_{ij}, \quad \bar{a}_{i_1, i_2, \dots, i_r} = \bar{b}_{i_1, i_2, \dots, i_r} \quad (14)$$

An interesting feature which distinguishes the operations in Section 2 from those in Section 3 is as follows. This is that, irrespective of the order of the tensor, the output for all the expressions in Section 2 is a scalar (tensor of order 0), while in Section 3 the output is of the same order as the order of the two entities included as inputs. Because of the fact that the tensor in the the of the fact that the order of the r.h.s is less than the order of the terms in the l.h.s., this feature is given the name "contraction" in standard tensor calculus. We shall generalize this for Boolean operations in the next section and make more precise definitions of contracted sums and products, Boolean sums and products, and also "expanded sums and products", of two Boolean tensors (which may or may not be of the same order). This is shown in the next two sections .

4. Contracted and expanded sums and products of vectors and tensors.

(a) Procedure of "contraction" in BVMF

Eq.(5) for the scalar product of two vectors effectively converts two tensors of order 1 into a tensor of order 0.

Similarly, Eq.(6) for the scalar triple product of two vectors (first order tensors) and a matrix (second order tensor) once again leads to a scalar, which is a tensor of order 0. As mentioned above, this procedure is well-known in standard tensor calculus, and the process of summing over repeated indices is known by the name "contraction". A general example of contracting two indices i and j is indicated in (14).

$$\sum_i \sum_j A_{ijk} \dots \ell B_{ij} = C_k \dots \ell \quad (14)$$

We give below, in (15a,b,c), some examples of this process in BVMF, which have been shown to be of particularly relevant to its applications for logic in our earlier reports.

Unary matrix relation : $\bigoplus_i a_i \otimes Z_{ij} = b_j \quad (15a)$

Binary matrix relation: $\bigoplus_j \bigoplus_i a_i \otimes R_{ijk} \otimes b_j = c_k \quad (15b)$

Tensor relation : $\bigoplus_i a_i \otimes R_{ijk} = b_{jk} \quad (15c)$

Clearly we can also have contractions via the Boolean sum operator as variations on this theme. — for instance,

$$\bigoplus_i \bigoplus_j A_{ijk} \dots l \otimes B_{ij} = C_k \dots l \quad (16a)$$

$$\bigotimes_i \bigotimes_j A_{ijk} \dots l \oplus B_{ij} = C_k \dots l \quad (16b)$$

Thus, we arrive at one type of general tensor product operation, (associated with the name "contraction", in standard calculus) as applied to the calculus in BVMF, in which the common indices of the two tensors that are contracted get eliminated and the remaining indices occur together on the r.h.s. The contraction process can be associated with either the Boolean sum or the Boolean product.

(b) "Expanded" direct product and direct sum in BVMF

In standard tensor calculus, the simplest example of an "expanded" product is the definition of certain types of tensors which can be obtained as a direct product (Cartesian product) of two vectors, namely

$$\underline{\underline{R}} = \underline{\underline{a}} \times \underline{\underline{b}} : a_i b_j = R_{ij} , \quad i = 1 \text{ to } m, j = 1 \text{ to } n \quad (17a)$$

More generally, we can have

$$\underline{\underline{B}} = \underline{\underline{a}} \times \underline{\underline{A}} : a_i A_{jkl} = B_{ijkl} \quad (17b)$$

It is to be noted that, as contrasted with contraction, the direct product produces an increase in the number of indices associated with the tensor, and the order of the tensor on the r.h.s is the sum of the orders of the two tensors on the l.h.s involved in the direct product. Remembering this criterion, we shall define the corresponding "expanded product" and "expanded sum" in BVMF and give a few examples. Thus, the "expanded product", in general, is defined by

$$A_i \dots j \cup B_k \dots l = C_i \dots jk \dots l \quad (18a)$$

and the general "expanded sum" is given by

$$A_i \dots j \oplus B_k \dots l = C_i \dots jk \dots l \quad (18b)$$

Some particular examples of these, which have been met with in SNS and quantifier algebra of predicate logic, are

$$a_i \otimes b_j = z_{ij}, \quad a_i \oplus b_j = z'_{ij}, \quad i = 1 \text{ to } m, \quad j = 1 \text{ to } n \quad (19a)$$

with $m = n = 2$ for the connectives \underline{A} and \underline{Q} of SNS standing for z_{ij} and z'_{ij} , and the corresponding operators in quantifier logic with $m = n = 3$. Also, in the application of BVMF to predicate calculus, generally we have found the need for operations of the type (20a,b) in Lecture-4, Series-2, MR-56.

$$c \otimes a_i = b_i, \quad c \oplus a_i = b_i \quad (20a)$$

These are special cases of the expanded sum and product — namely the expansion of a tensor of order 0 with a tensor of order 1, leading to a tensor of order 1 ($= 1 + 0$) on the r.h.s.

More generally, the formulation of logical tensor relations can be given on the basis of the expanded product of r individual vectors leading a tensor of order r . Two examples are

$$a_i \quad b_j \quad c_k = \mathbb{R}_{ijk} \quad (21a)$$

$$((a_i \otimes b_j) \oplus c_k) \otimes d_l = \mathbb{R}_{ijkl} \quad (21b)$$

In the form of (21a), the Boolean truth value of the relation specified by the tensor \mathbb{R}_{ijk} , for input vectors u_i, v_j, w_k , is given by (22)

$$\bigoplus_k \bigoplus_j \bigoplus_i \mathbb{R}_{ijk} \otimes u_i \otimes v_j \otimes w_k = \langle a_i | u_i \rangle \otimes \langle b_j | v_j \rangle \otimes \langle c_k | w_k \rangle \quad (22)$$

This is only given as an illustration of the simplification of a general tensor relation \mathbb{R} when it is expressible as the expanded product of r vectors. This is particularly employed in our formulation of SNS and QL-2 algebra as indicated in (19a,b).

